# UNIVERSIDAD AUTONOMA DE MADRID

## ESCUELA POLITECNICA SUPERIOR



**Grado en Ingeniería Informática**

# TRABAJO FIN DE GRADO

## SISTEMA DE CONTROL DE ASISTENCIA A CLASE DE FORMA NO INTRUSIVA

**Javier Quinzaños Baena**
**Tutor: Fernando Maestre Miranda**
**Ponente: Álvaro Ortigosa Juárez**

**Junio 2018**

# SISTEMA DE CONTROL DE ASISTENCIA A CLASE DE FORMA NO INTRUSIVA

**AUTOR: Javier Quinzaños Baena**
**TUTOR: Fernando Maestre Miranda**
**PONENTE: Álvaro Ortigosa Juárez**

**Dpto. Ingeniería Informática**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Junio de 2018**

# Resumen

En la actualidad se está viviendo una transformación digital en la cual muchas de las herramientas cotidianas están siendo sustituidas por diversas aplicaciones móviles o sistemas hardware/software. Un ejemplo de esto pueden ser los mapas, que han sido sustituidos por aplicaciones o aparatos GPS que calculan automáticamente la mejor ruta entre dos puntos y muestran lugares de interés cercanos.

Este Trabajo Fin de Grado vamos a diseñar un sistema software mediante el cual poder sustituir las anotaciones manuales de asistencia a clase que toman los profesores de la UAM a través de una aplicación móvil que permita introducir y visualizar las asistencias de los alumnos. En ella los profesores podrán activar o desactivar el registro de asistencias y de este modo permitir o no a los alumnos marcar su asistencia para esa clase. Además, permitirá, tanto a alumnos como a profesores, visualizar el porcentaje de faltas de asistencia a cada una de sus asignaturas, en el caso del profesor vería los porcentajes de todos sus alumnos.

Para realizar este proyecto se han empleado una amplia variedad de tecnologías y herramientas, como son:

1. SQL y servidores de base de datos en la nube para almacenar la información recogida por el sistema.
2. Python para el desarrollo de un script automático encargado de introducir información en la base de datos.
3. Archivos JSON para guardar, tanto diversos parámetros de configuración del script, cómo para contener la información necesaria por la aplicación (horario y aulas de cada asignatura, alumnos y profesores de la UAM, clases en las que intervienen e información similar).
4. Las diversas herramientas del framework Xamarin para el desarrollo de la aplicación multiplataforma bajo el lenguaje de programación C# y XAML.

Esto nos ha permitido aprender mejor el uso de distintas tecnologías de manera conjunta, así como descubrir nuevas herramientas tan versátiles como el framework Xamarin.

El sistema desarrollado ha sido satisfactorio y ha cumplido con éxito los requisitos planteados inicialmente. Para probar que esto era así se realizaron diversas pruebas en distintos momentos del día con clases simuladas en las que había varios alumnos y un profesor.

# Palabras clave

Xamarin, aplicación, multiplataforma, móvil, dispositivo, sistema, SQL, Python, base de datos, servidor, Azure, asistencia, asignatura, profesor, alumno

# Abstract

Currently a digital transformation is underway in which many of the daily tools are being replaced by several mobile applications or hardware/software systems. An example of this can be maps, which have been replaced by applications or GPS devices that automatically calculate the best route between two points and show nearby places of interest.

In this Bachelor Tesis we will design a software system to replace the manual annotations of class attendance taken by UAM professors through a mobile application that allows to introduce and visualize the attendance of students. In it, the teachers can activate or deactivate the register of attendances and, in this way, allow or not students to mark their attendance for that class. It will also allow both students and teachers to see the percentage of absences of attendance to each of their subjects, in the case of the teacher would see the percentages of all their students.

A wide variety of technologies and tools have been used to carry out this project. SQL and database servers were used in the cloud to store the information collected by the system, such as:

1. Python for the development of an automatic script responsible for entering information in the database.
2. JSON files to save configuration parameters of the script and to contain the information needed by the application (hours and classrooms of each subject, students and professors of the UAM, classes in which they took part and information like that)
3. Various Xamarin framework tools for the development of the multiplatform application using C# programming language an XAML for the design of user interfaces.

This has allowed us to better learn the use of different technologies working together, as well as discover new tools as versatile as the Xamarin framework.

The system developed has been satisfactory and has successfully met the requirements initially set. To prove that this was the case, several tests were carried out at different times of the day with simulated classes in which there were several students and a teacher.

# Keywords

Xamarin, application, multiplataform, smartphone, device, system, SQL, Python, database, server, Azure, attendance, subject, teacher, student

## *Agradecimientos*

Debo dar las gracias a Francisco de Borja por la idea de usar las MACs de los dispositivos cercanos para un sistema de control de asistencia.

A Fernando Maestre por su sistema personal de control de asistencia a clase, el cual me sirvió de inspiración.

Y especialmente a Álvaro por sus buenas ideas para facilitar el desarrollo, testeo y validación de la aplicación.

# INDICE DE CONTENIDOS

# INDICE DE FIGURAS

# INDICE DE TABLAS

# 1 Introducción

El objetivo de este Trabajo de Fin de Grado es desarrollar un sistema de control de asistencia a clase no intrusivo, es decir, que sea más cómodo y rápido de usar que los métodos tradicionales, pero igualmente efectivo.
Para ello se desarrolló una aplicación multiplataforma, haciendo uso del framework Xamarin [1], con la cual marcar las asistencias de los alumnos y poder visualizar el número de faltas de estos
Además, se usa un ordenador de bajo coste como factor de verificación, el cual se encarga de ubicar a los alumnos en las aulas mediante los mensajes DHCP enviados por los clientes WIFI.
A continuación, se explican los objetivos y la motivación de este trabajo, y se describe la organización de este documento.

## *1.1 Motivación*

Actualmente disponemos de gran cantidad de aplicaciones móviles con funcionalidades muy variadas, muchas de ellas se han convertido en herramientas de nuestra vida diaria que nos facilitan nuestras actividades cotidianas. También estamos observando la sustitución de los elementos tradicionales en aras de la tecnología, muchas veces siendo suplidos por diversas aplicaciones móviles como en el caso de los mapas físicos por Google Maps [2] y se está empezando a pagar mediante NFC [3] con Google Pay [4] o aplicaciones móviles tipo Bizum [5].
Además de estas aplicaciones también existen otras orientadas a la educación. Algunas de ellas te permiten ver tus calificaciones, las tareas que tienes que realizar y el temario a estudiar. Otras ayudan en la planificación del estudio o para organizar el horario de cada día. Nuestra aplicación entrará también en esta categoría.

Actualmente, para pasar lista en las clases se suele usar papel y bolígrafo, bien sea el profesor que va nombrando y tachando los alumnos, o bien los alumnos que firman junto a su nombre. Este proceso puede resultar lento y engorroso, además de que se puede traspapelar alguna hoja y que los datos recogidos no son tan fácilmente manejables como tenerlos informatizados. Además de esta forma el alumno no lleva un control de las asistencias que vaya sincronizado con la información del profesor, con lo cual se pueden dar casos de que el profesor piense que el alumno faltó más y éste no esté de acuerdo.
Estos problemas hacen que a la hora de calificar o revisar las asistencias falten datos o sea más engorroso de obtener la información deseada, por ejemplo, para ver el detalle de un día tienes que buscar la hoja de ese día entre cientos de hojas de esa asignatura, y ni que decir tiene el engorro que puede ser si tienes varias asignaturas.
Es por ello por lo que se plantea desarrollar un sistema informático que permita sustituir las listas en papel, y así poder llevar un control más preciso y cómodo de usar.

## 1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado será desarrollar un sistema informático que permita gestionar las asistencias a clase, así como consultar las faltas de cada alumno a las diversas asignaturas, de modo que se puedan sustituir las listas impresas en las que se firma o se toman anotaciones manuales.
Como paso previo a la realización del proyecto y como se verá más adelante en esta memoria, se realizó una investigación sobre las diversas tecnologías actualmente existentes para desarrollo de aplicaciones móviles. Como resultado de la investigación realizada, se descubrió una herramienta que no había sido explorada anteriormente por el autor del trabajo. Esta herramienta es el framework Xamarin y como se comentará más adelante permite crear aplicaciones móviles multiplataforma que son compiladas de forma nativa para cada plataforma, haciendo uso de gran parte de código común, tanto de interfaz como de funcionalidad.
Se considerará que los objetivos se han cumplido y que el proyecto ha terminado exitosamente si se consigue una aplicación móvil que sea funcional y que pueda sustituir a los medios tradicionales empleados actualmente.

## 1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte:** en este capítulo se analizarán aplicaciones similares que puedan servir como punto de partida o como orientación a la hora de comenzar nuestro proyecto.
- **Definición del proyecto:** aquí se estudiará el alcance de la aplicación, la metodología, herramientas y tecnologías utilizadas en su desarrollo. Además, se realizará una recopilación de los requisitos funcionales y no funcionales que debe cumplir nuestro proyecto.
- **Diseño:** en este apartado se hablará de cómo se estructura el sistema desarrollado en cuanto a los componentes utilizados y su interacción, del diseño de la aplicación y de algunas características esenciales.
- **Desarrollo:** en este apartado abordaremos la forma en la que se ha llevado a cabo el proyecto una vez finalizada la fase de diseño. Tratará de todo el proceso de codificación y las eventualidades que haya habido en él.
- **Pruebas y resultados:** durante el desarrollo del proyecto se realizaron pruebas de las distintas partes que se iban creando, así como pruebas finales una vez estuvo listo el primer prototipo. Cuando se consideró la aplicación finalizada, se realizaron pruebas de validación y se utilizó en el ámbito real donde se espera que sea utilizada. Todas estas pruebas y sus resultados se comentarán en este apartado de la memoria.
- **Conclusiones y trabajo futuro:** para finalizar el cuerpo de este documento, se comentarán las conclusiones que se pueden elaborar tanto de la aplicación en sí como del trabajo realizado. A su vez, se comentarán posibles mejoras que se puedan hacer y el trabajo futuro que se puede llevar a cabo.
- **Referencias:** aquí se incluye la bibliografía y referencias que se han tenido en cuenta a la hora de documentar el proyecto y elaborar esta memoria.

- **Glosario:** se incluyen las palabras más importantes en este proyecto y una breve definición de estas.
- **Anexos:** entre ellos se encuentran un manual de instalación de la aplicación en un dispositivo Android, un manual para el programador con una explicación para la utilización del Framework Xamarin, y algunas capturas de pantallas correspondientes a las ventanas principales de la aplicación.

# 2 Estado del arte

## 2.1 Introducción

Desde el lanzamiento de la Play Store de Google el 28 de octubre de 2008 [6] y de la App Store de iOS el 10 de julio de 2008 [7] se han lanzado gran cantidad de aplicaciones para dispositivos móviles Android e iOS en sus respectivas plataformas, que han sufrido un gran crecimiento tanto en número de aplicaciones como en número de desarrolladores desde su creación.
La evolución es muy notable en ambas tiendas de aplicaciones: Play Store tenía 2.300 aplicaciones en marzo de 2009 [6] y para marzo de 2018 ese número aumentó hasta los 3,8 millones [8], y en el caso de App Store, en su creación en julio de 2018 disponía de 500 aplicaciones para llegar en marzo de 2018 hasta los dos millones [8].
Además de estas dos tiendas de aplicaciones, que son las que mayor cantidad tienen, existen otras de menor importancia como pueden ser Windows Store, Amazon Appstore y Blackberry world; que en marzo de 2018 contaban con 669.000, 430.000 y 234.500 aplicaciones disponibles para descarga respectivamente [8].
Con estas cifras tan numerosas podemos encontrar prácticamente cualquier aplicación que nos imaginemos, incluso muchas veces disponemos de varias alternativas que responden a una misma necesidad, teniendo que investigar para escoger la que mejor nos convenga.

## 2.2 Aplicaciones relacionadas con la educación

Últimamente se está dando un cambio en el sector educativo incorporando más tecnología en las aulas, como pueden ser las pizarras electrónicas, tablets en las que llevar los libros, donde realizar exámenes o tomar apuntes, y los proyectores con los que hacer presentaciones o demostraciones en tiempo real.

Existen multitud de aplicaciones móviles centradas en la educación, cada una con una funcionalidad o características concretas. Unos ejemplos son:

- *Doulingo:* esta aplicación permite aprender varios idiomas a través de juegos, cuestionarios, videos y audios de forma gratuita. [9]
- *Agenda Escolar:* con ella puedes controlar tu horario, tomar notas de las tareas que tienes que realizar y recibir notificaciones sobre ellas, almacenar la información de contacto de los profesores y apuntar las calificaciones para acceder a ellas y calcular la media. [10]
- *Udemy:* es una plataforma de cursos en línea gratuitos (MOOCs) a los que puedes acceder a través de la web o de esta aplicación. [11]
- *Kahoot:* se trata de una aplicación/web que permite crear cuestionarios personalizados y con puntuación útil para preguntas o controles breves durante una clase. [12]
- *LiveBoard:* esta aplicación permite compartir entre varias personas un área donde escribir, tomar notas, apuntar ideas, etc. Puede ser útil para trabajos en equipo. [13]

Ya hemos visto que se dispone de muchas y variadas opciones, pero aún no se ha implantado un sistema que permita el control de asistencia, más allá del método tradicional de papel y bolígrafo.

En este Trabajo de Fin de Grado vamos a desarrollar un sistema que nos permita recoger y controlar las asistencias de los alumnos de forma fiable y rápida. Como referencia vamos a buscar en la Play Store aplicaciones que sirvan para el mismo objetivo, para ello buscamos la palabra clave "attendance" y podemos observar el resultado en la Figura 2-1.

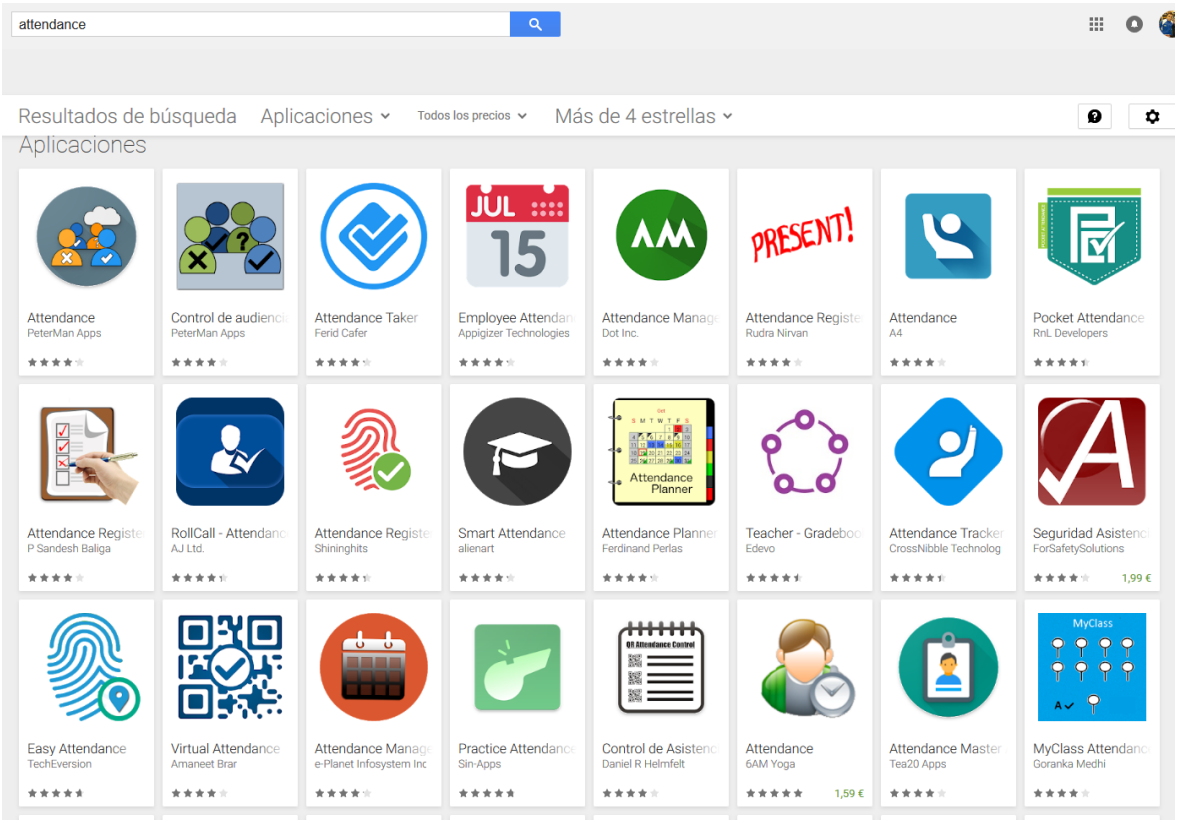


**Figura 2-1: Resultados de la búsqueda de la palabra "attendance" en la Play Store**

Sobre estos resultados vamos a elegir las cuatro aplicaciones más similares a lo que queremos hacer y las vamos a comparar para ver las diferencias entre ellas y lo que debe tener la aplicación que vamos a desarrollar. De esta forma podremos saber que aspectos debemos imitar, cuales hay que mejorar o qué cosas nuevas podemos aportar.

Los ejemplos que vamos a tomar como base son:

- ***Attendance*** [14]**:** se trata de una aplicación que nos permite controlar la asistencia de personas a distintos tipos de eventos. Tiene dos inconvenientes o una perspectiva distinta a lo deseado: la primera es que los participantes no pueden ver sus asistencias sincronizadas con las del creador del evento, y la segunda es que tanto el evento como los participantes se añaden a mano (en el caso de participantes se pueden importar desde un archivo).

Para indicar las horas y días en los que tiene lugar el evento se añaden instancias del evento manualmente, pudiéndole indicar frecuencia semanal durante un periodo para introducir todas de una vez. Después la asistencia se marca sobre cada instancia por separado.

- ***Attendance Register (students)*** [15]**:** esta aplicación va destinada a que los estudiantes controlen sus asistencias. En este caso la forma de visualizar e introducir las asistencias es a través de un calendario, el cual marca en verde las asistencias y en rojo las faltas, y al pulsar sobre un día te permite decir si asististe o no. Además, permite añadir notas.
  Tiene los mismos inconvenientes que la anterior, con el añadido de que solo puede haber una clase de una misma asignatura en el mismo día, y no está definida la hora a la que empieza.
- ***Pocket Attendance*** [16]: al igual que las dos anteriores, esta aplicación va destinada a que los estudiantes lleven un control de sus asistencias. Tiene los mismos inconvenientes comentados anteriormente: no va sincronizada la información con la del profesor, y las clases se añaden manualmente.
- ***Smart Attendance*** [17]: en este caso va destinada a que los profesores controlen las asistencias de los alumnos. Actualmente no funciona adecuadamente, ya que no se pueden ver las asistencias añadidas.

| **Aplicación** | **Introducción de asignaturas y alumnos automática** | **Registro de asistencias (por parte del alumno)** | **Registro de asistencias (por parte del profesor)** | **Visualización de asistencias (por parte del alumno)** | **Visualización de asistencias (por parte del profesor)** | **Permite varias clases de la misma asignatura por día** | **Verificación extra de asistencia** |
|---|---|---|---|---|---|---|---|
| Attendance | No | A medias | Sí | A medías | Sí | Sí | No |
| Attendance Register (students) | No | Sí | No | Sí | No | No | No |
| Pocket Attendance | No | Sí | No | Sí | No | Sí | No |
| Smart Attendance | No | No | Sí | No | No funciona | Sí | No |

**Tabla 2-1: comparativa de funcionalidad de aplicaciones similares**

La aplicación que se va a desarrollar debe tener las características indicadas en la tabla anterior, con el matiz de que el profesor no va a registrar las asistencias, sino que va a activarlas para que los alumnos puedan indicar que están presentes, o desactivarlas para que no puedan seguir marcando su asistencia, de este modo se tiene mejor control de los días que ha habido clase y del tiempo durante el cual el profesor quiere que se pueda registrar la asistencia de los alumnos.

De las cuatro aplicaciones tomadas como ejemplo se podría decir que la más similar a lo que se pretende hacer es *Attendance* pero esta aplicación no aporta el extra de seguridad que tiene el uso de las MACs como verificación de que el alumno se encuentre físicamente en el aula, ni introduce automáticamente las asignaturas que imparte cada profesor o en las que se encuentra matriculado cada alumno, además de que los datos introducidos por el alumno y el profesor no están sincronizados de ninguna forma.

## *2.3 Sistemas de control de asistencia*

Debido al cambio que se está experimentando en el sector educativo, y a la necesidad de controlar de formas más fiables las asistencias de tanto los alumnos como los profesores están apareciendo sistemas de control como el que queremos desarrollar mediante este Trabajo de Fin de Grado.

Un ejemplo de ello es el sistema desarrollado por la Universidad Pontificia de Salamanca [18], el cual se basa en chips o etiquetas NFC (Near Field Communication). El funcionamiento de este sistema es el siguiente: cada alumno dispone de una etiqueta, al comienzo de cada clase el alumno la acerca a un lector, o al dispositivo del profesor que actúa como lector, y de esa forma se registra la asistencia. Luego, tanto el alumno como el profesor disponen de una aplicación en la cual poder visualizar las asistencias.

En este caso, observamos muchas similitudes con el objetivo de este Trabajo de Fin de Grado: se dispone de un sistema de seguridad extra debido al uso del NFC (verificando que se encuentre físicamente en alumno en la hora de clase al tener una etiqueta única y tener que escanearla en el dispositivo del profesor), la asistencia la determina el alumno en lugar del profesor, y, por último, tanto el alumno como el profesor comparten y visualizan la misma información (en caso del profesor, la de todos los alumnos, y para el alumno, solo la suya).

La principal diferencia radica en el mecanismo extra de seguridad implementado: en nuestro caso vamos a usar las MACs de los dispositivos cercanos y el otro sistema usaba NFC. No hay diferencias significativas entre usar uno u otro, quizás sea más complicado para los alumnos hacer trampas en el caso del NFC ya que, al tener el profesor el dispositivo, un alumno no va a poder pasar la tarjeta de un compañero, ni que por el rango de detección del dispositivo que captura las MACs pueda marcar que ha asistido desde el pasillo o la clase de al lado.

Otro ejemplo de un sistema de control de asistencia a clase es el que implementó el tutor de este Trabajo de Fin de Grado, Fernando Maestre. El funcionamiento de este sistema se basa en una web sobre la cual se sustenta la aplicación, primero el alumno debe iniciar sesión con su cuenta (que había sido creada previamente en otro apartado de la aplicación), luego el profesor habilita la asistencia y asigna números en orden a todos los alumnos de la clase (y él se asigna el número sucesivo al último alumno), entonces cada estudiante introduce el número que le fue asignado y pulsa un botón para registrar la asistencia, y por último, el profesor cierra el registro.

En este otro caso también podemos observar similitudes, no tanto en el soporte ni en la forma de tomar asistencias, pero sí en la necesidad de registro, la identificación de los alumnos y el mecanismo extra de seguridad (al tener bien identificados a los alumnos, y asignar un número a cada uno teniendo el profesor el último, no puede haber ningún alumno que no esté en la clase, pero sí haya marcado su asistencia). Es otra forma de implementación igualmente válida, pero lo que se persigue con este Trabajo de Fin de Grado es hacer algo más automático y que englobe distintas tecnologías.

## *2.4 Conclusiones*

Como hemos podido observar hay muchas aplicaciones relacionadas con la educación. Dentro de esa categoría podemos encontrar bastantes de ellas dedicadas al control de asistencia similares a lo que queremos hacer, pero ninguna cumple todas las características deseadas. En concreto, ninguna es capaz de introducir automáticamente las asignaturas ni los alumnos (esto se debe a que su objetivo es ser genéricas, y alguna de ellas si permite cargar esta información desde un archivo), así como tampoco disponen de un sistema extra que corrobore que el alumno se encuentra físicamente en el aula. Además, hay otras que cumplen parcialmente los requisitos o no funciona correctamente.

En este Trabajo de Fin de Grado se realizará una aplicación que solucione estos problemas y que disponga de la funcionalidad completa. Así como también realizaremos suficientes pruebas para garantizar que la aplicación sea precisa (contabilice correctamente las asistencias) y que no falle durante su uso en ninguno de los sistemas operativos para los cuales debe operar la aplicación.

# 3 Análisis y Definición del proyecto

## *3.1 Introducción*

En este apartado de la memoria se definirá el proyecto y lo que se va a realizar en este Trabajo de Fin de Grado. En concreto se estudiará su alcance, la metodología, tecnologías y herramientas empleadas durante su desarrollo y se detallaran tanto los requisitos funcionales como los no funcionales.

## *3.2 Alcance*

Este Trabajo de Fin de Grado tiene el propósito de desarrollar un sistema de control de asistencia a clase que pueda ser usado tanto por alumnos como por profesores para llevar un control de las asistencias o faltas a clase que sea sencillo, rápido y seguro de usar.

Fuera del alcance quedará la exportación y explotación de la información, así como el poder añadir o eliminar asignaturas o alumnos desde la aplicación. Es decir, el sistema a desarrollar solo contemplará el recopilar y visualizar las asistencias o faltas de los alumnos.

## *3.3 Metodología*

Para el desarrollo del sistema se siguió una metodología ágil [19] que permitiese incorporar o modificar funcionalidades durante el desarrollo cómo resultado de la fase de pruebas de cada prototipo o cómo una nueva necesidad detectada.
El desarrollo ágil es iterativo e incremental. En él se realizan iteraciones del ciclo de vida en las cuales se realizan todas las fases del proyecto (análisis, diseño, codificación y pruebas) para dar lugar a un producto listo para ser usado, conocido como *prototipo.* En las siguientes iteraciones se repite este proceso pudiéndose modificar los requisitos previos y llegando a un prototipo con mayor funcionalidad, hasta que en la última iteración se llegue al producto final.

Las diferentes fases que se han seguido en el desarrollo han sido las siguientes:

1. **Definición del proyecto y su alcance:** en primer lugar, se acotó la idea principal de este Trabajo de Fin de Grado, estableciendo de este modo su alcance y los objetivos a alcanzar al finalizarlo.
2. **Análisis de requisitos:** tras la definición del proyecto debemos centrarnos en definir los requisitos funcionales y no funcionales que debe cumplir el sistema para que este se considere exitoso.
3. **Investigación de las herramientas y tecnologías a usar:** una vez claros los requisitos realizamos una investigación para ver las herramientas disponibles para el desarrollo.
4. **Diseño del sistema:** comenzamos haciendo un diagrama de interacción entre componentes y bocetos para el diseño de la aplicación.
5. **Codificación:** se comienza la programación de tanto la aplicación como del script que va a capturar las MACs teniendo en cuenta los requisitos y el diseño realizado en las anteriores etapas.

6. **Pruebas unitarias:** por cada componente del sistema se comprueba que cumpla los requisitos establecidos. Si alguna de ellas fallase se volvería a la etapa anterior para corregirlo y volver a probar hasta solucionarlo.
7. **Pruebas de validación y verificación:** tras haber finalizado el desarrollo del sistema, se comprueba que el mismo funcione adecuadamente en el entorno destino corrigiendo los errores encontrados o mejorando la usabilidad. De este modo se observa un ciclo entre las fases 5, 6 y 7 tal y como tiene lugar en las metodologías agiles que se están siguiendo en este proyecto.
8. **Pruebas de compatibilidad:** debido a que estamos desarrollando una aplicación multiplataforma, se debe garantizar que el sistema funcione correctamente en todos los sistemas operativos deseados, así como con diversos tamaños de pantalla.
9. **Puesta en marcha del sistema:** una vez completadas exitosamente todas las pruebas, el sistema entrará en producción y se utilizará en lugar de las anotaciones manuales.

El ciclo de vida terminaría con una última fase de mantenimiento del sistema desarrollado, en el cual habría que solucionar los fallos que se detecten o modificarlo para cumplir nuevos requisitos. Este mantenimiento ya no sería objeto de este Trabajo de Fin de Grado, sino que se realizaría una vez que el sistema se encuentre en funcionamiento.

## *3.4 Tecnologías y herramientas*

Para implementar el sistema propuesto en este Trabajo de Fin de Grado se necesitaban tres piezas, que son las siguientes:

- La primera de ellas es un miniordenador que nos permitiese escanear y guardar en una base de datos en la nube la información, que fuese de bajo coste, ya que tendrá que instalarse uno por aula para garantizar la localización del estudiante en cada aula que tenga clase.
  Para ello se usó una Raspberry Pi [20], en concreto el modelo 3B, debido a que ya se disponía de una, el precio que tiene es sobre los 35 o 40 euros (solo la placa, en pack sobre los 50 o 60 euros), pero existe una versión más barata que podría ser usada en su lugar, la Raspberry Pi Zero W que tiene un precio entre 11 y 15 euros (solo la placa, en pack sobre los 25 euros).
- La siguiente pieza obvia es, por tanto, la propia base de datos. En ella almacenaremos tanto la información de asistencias, como diversa información de utilidad para el sistema (como los datos de sus usuarios y otra información de interés). Estará ubicada en la nube, en concreto, en un servidor de Microsoft Azure, el cual permite escalabilidad automática en función de volumen de datos y peticiones, en caso de usar un plan de pago.
- Y la segunda pieza es una aplicación móvil con la cuál poder confirmar y visualizar las asistencias. Este proyecto quería desarrollarse de modo que fuese compatible con varias plataformas, para que de este modo pudiese ser usado por todos los estudiantes y profesores, es decir, que se pueda ejecutar bajo Android [21], iOS [22] y Windows [23], que son las tres plataformas más usadas actualmente.

Tras una primera fase de investigación, fue descubierto el framework Xamarin para el desarrollo de la aplicación. Esta herramienta permite desarrollar aplicaciones móviles multiplataforma con solo programar una versión, es decir, se diseña una única vez la interfaz y se programa el comportamiento que debe tener de forma genérica, y una vez hecho eso esta herramienta se encarga de compilar el código de forma nativa en cada plataforma. En caso de que se necesite hacer algo especial en cada plataforma, o que el framework no proporcione una interfaz genérica, se puede especificar el comportamiento de un fragmento de código para la plataforma o plataformas que se necesite.
La decisión de usar esta herramienta vino dada por la necesidad de desarrollar una aplicación lo más compatible posible, así como por la curiosidad de aprender una nueva herramienta tan versátil.

A continuación, se detallan las tecnologías y herramientas usadas en el desarrollo.

### 3.4.1 Tecnologías

Para la parte de la Raspberry se usaron las siguientes tecnologías:

- Python [24]: es un lenguaje de programación multiparadigma (soporta orientación a objetos, programación imperativa y programación funcional) con tipos dinámicos, es multiplataforma y es interpretado. Además, es ampliamente utilizado para pequeños scripts. En este caso se usa tanto para llamar al programa que se encarga de capturar las MACs como para introducir los resultados, una vez procesados, en la base de datos.
- Bash scripting [25]: Bash es un tipo de shell de Linux que permite ejecutar scripts de código. En este proyecto se usa para llamar a un ejecutable desde el script Python de una manera más simple que haciéndolo directamente en Python.
- Microsoft SQL Server [26]: se trata de un sistema de manejo de bases de datos creado por Microsoft, el cual hace uso de una implementación del estándar ANSI del lenguaje SQL (llamada Transact-SQL, TSQL). En nuestro caso hemos usado Azure SQL, que es una implementación en la nube de Microsoft de un servidor SQL Server, para almacenar la información de las asistencias de los alumnos.
- Mensajes DHCP Discovery sobre WiFi [27]: este tipo de mensajes se envía periódicamente por los clientes para localizar las distintas redes y las direcciones de sus servidores DHCP para poder realizar la conexión, a estos paquetes también se les denomina *Beacons*. En este sistema se usan como mecanismo de identificación de estudiantes, capturamos todos los paquetes que circulan cerca del receptor (la Raspberry Pi) de cada clase extrayendo e introduciendo en la base de datos cada una de las MACs detectadas.

Para la aplicación hicimos uso de estas tecnologías:

- C# [28]: se trata de un lenguaje de programación multiparadigma (orientación a objetos, imperativa y funcional) que inicialmente se podía ejecutar únicamente en Windows, pero con el desarrollo del compilador Mono, puede ser ejecutado en prácticamente todas las plataformas.

Este lenguaje de programación es usado por el framework Xamarin para el desarrollo del código compartido.

- XAML [29]: se trata de un lenguaje de formato utilizado para diseñar interfaces de usuario independientes de la plataforma, el funcionamiento es similar al del HTML, pero la sintaxis es la del XML con etiquetas y atributos nuevos. Es usado por el framework Xamarin para el diseño de la interfaz.
- JSON [30]: se trata de un formato de texto ligero para intercambio de datos. En este proyecto se usa para almacenar la información de alumnos, profesores y aulas, junto con los horarios de las clases, dentro de la aplicación y leerla de forma rápida.
- Microsoft SQL Server: en este caso se usa tanto para insertar las asistencias como para comprobar distinta información del usuario y de las asistencias.

Y, como no podía ser de otra manera, para el desarrollo de la base de datos se usó el lenguaje SQL, siguiendo un dialecto propio de SQL Server denominado Transact-SQL.

### 3.4.2 Herramientas

Para el desarrollo y codificación hemos usado las siguientes herramientas:

- Xamarin: es un framework gratuito y open source que permite realizar aplicaciones móviles multiplataforma mediante C# para la codificación y XAML para el diseño de la interfaz.
- Visual Studio [31]: es un IDE (Entorno de Desarrollo Integrado) para sistemas operativos Windows, y recientemente MacOS, que soporta gran variedad de lenguajes de programación y plugins para aumentar su funcionalidad. Dispone de integración con Xamarin para compilar, depurar, y ejecutar emuladores en los que probar la aplicación desarrollada, haciendo más fácil y cómodo el desarrollo.
- Atom [32]: es un editor de texto que marca la sintaxis de gran variedad de lenguajes de programación, y es altamente personalizable y extensible mediante plugins. Se usará para escribir el código Python y Bash, así como para visualizar rápidamente tanto los cambios (debido a que integra Git) como ciertos archivos de la aplicación.
- DataGrip [33]: se trata de un IDE multiplataforma que permite conectarse y gestionar distintos sistemas de Bases de Datos, en concreto lo usaremos para conectarnos al servidor Azure en el que almacenamos la base de datos de la aplicación, crear las tablas y ver los datos que se van introduciendo.

Para el control de versiones estamos haciendo uso de las siguientes herramientas:

- Git [34]: es un software diseñado para dar soporte al control de versiones ampliamente utilizado en proyectos software.
- Bitbucket [35]: es una plataforma gratuita que hace uso de Git y que permite crear repositorios privados en la nube, gestionando de manera más sencilla los cambios y las distintas versiones posibles.

- Visual Studio Team Foundation: se trata de una funcionalidad de Visual Studio que integra Git con el IDE, permitiendo visualizar y subir los cambios a un repositorio remoto de Git (y de otros sistemas de control de versiones).

Para realizar las pruebas se utilizaron diversos emuladores Android para ver de forma rápida cómo iba quedando la interfaz de la aplicación y probar su correcto funcionamiento. Además, se realizaron pruebas en dispositivos físicos como son un smartphone BQ Aquaris X5 con Android 7.1.2 y una tablet Asus Transformer con Android 6.0.
Al no disponer físicamente de teléfonos móviles con iOS ni Windows las pruebas en estas plataformas no se han podido realizar, pero no debería haber ningún problema al estar haciendo uso de un framework multiplataforma nativo.

En cuanto al uso de bibliotecas o librerias, únicamente usamos las propias del lenguaje de programación y del framework, así como de NewtonSoft JSON [36] para la lectura de los ficheros JSON que contienen la información de alumnos, profesores y aulas, junto con sus horarios.

## *3.5 Requisitos*

En este subapartado se definirán los requisitos que debe cumplir el sistema y las funcionalidades de las que debe disponer, estos requisitos se dividirán en dos grupos: los requisitos funcionales (toda funcionalidad que deba tener) y los requisitos no funcionales (requisitos de operación, calidad y usabilidad).

### 3.5.1 Requisitos Funcionales

**RF 1:** Los usuarios han de registrarse en la aplicación para estar identificados. Los profesores deberán indicar únicamente su correo electrónico de la UAM. En el caso de los alumnos, además tendrán que introducir la dirección MAC de su dispositivo, para ello la aplicación abrirá la parte correspondiente a los ajustes WiFi del dispositivo, donde el alumno podrá localizar su MAC y copiarla para introducirla.

**RF 2:** Habrá dos tipos de usuarios: los alumnos, y los profesores. El tipo de usuario se elegirá al momento de registrarse en la aplicación.

**RF 3:** Se debe verificar que realmente es alumno o profesor de la UAM.

**RF 4:** Los profesores deben poder activar y desactivar las asistencias de cada clase, de modo que los alumnos puedan o no marcar su asistencia.

**RF 5:** Los alumnos podrán marcar su asistencia una única vez por clase y solo si el profesor ha abierto las asistencias. Esto solo se puede realizar si en la hora que marca el dispositivo del alumno existe una asignatura en la que está matriculado el alumno, si no existe debe informar de este hecho.

**RF 6:** Los profesores deben poder ver, para cada una de las asignaturas que imparten, el número o porcentaje de faltas de cada alumno.

**RF 7:** Los alumnos deben poder ver, para cada una de las asignaturas en las que está matriculado, el número o porcentaje de faltas.

**RF 8:** El sistema capturará y almacenará continuamente, en intervalos de un minuto, todas las MACs, que no haya visto anteriormente, cercanas al dispositivo instalado en cada aula de modo que una vez el alumno que tiene esa MAC quiera confirmar la asistencia se cumpla la verificación extra de que estuviese presente en el aula.

### 3.5.2 Requisitos No Funcionales

**RNF 1:** La aplicación debe ser compatible con varias plataformas, en concreto, Android, iOS y Windows.

**RNF 2:** La aplicación funcionará y se adaptará adecuadamente tanto en smartphone como en Tablet.

**RNF 3:** La aplicación hará uso de la pantalla táctil del dispositivo como método de interacción con el usuario.

**RNF 4:** La aplicación necesitará conexión a Internet para su correcto funcionamiento, ya que debe hacer consultas a una base de datos en la nube.

**RNF 5:** Los usuarios (alumnos y profesores) vendrán definidos en un archivo JSON que irá empaquetado dentro de la aplicación, es decir, se debe redesplegar la aplicación tras ser modificado. La lista de asignaturas tanto del alumno como del profesor se encontrará dentro de este archivo, junto al usuario que las tiene.

**RNF 6:** Los alumnos que asisten a una asignatura se encontrarán en otro archivo JSON, al igual que el caso anterior, dentro de la aplicación.

**RNF 7:** El horario y la ubicación de las asignaturas en las aulas vendrá definido en otro archivo JSON de la misma manera que los anteriores casos.

**RNF 8:** La información detallada de las asignaturas (nombre y profesor que la imparte) se encontrará en otro archivo JSON igual que se mencionó en los anteriores puntos.

**RNF 9:** El dispositivo instalado en cada aula para capturar las MACs cercanas ha de ser de bajo coste y con requisitos energéticos bajos, es decir, que sea barato y gaste poca energía. Así mismo, debe ser de pequeño tamaño para poder ser colocado fácilmente en cualquier lugar del aula y que no estorbe.

# 4 Diseño

## 4.1 Introducción

En este capítulo se tratarán las decisiones de diseño tomadas para desarrollar la aplicación definida en el anterior capitulo. En concreto estudiaremos cómo está estructurado el sistema y cómo se comunican o interaccionan las distintas partes de este, así como se explicará la arquitectura y el diseño de la aplicación tanto del código como de la interfaz.

## 4.2 Arquitectura del sistema

El sistema que se va a desarrollar en este Trabajo de Fin de Grado consta de tres piezas fundamentales, como ya se comentó en el alcance:

- Una Raspberry Pi por aula en la que se quiere tomar asistencias mediante este sistema. Su labor consiste en escuchar todos los paquetes DHCP Discovery que circulan a su alrededor, capturando e introduciendo a través de consultas SQL las distintas MACs encontradas. El uso de una Raspberry Pi para esta labor se debe al *RNF 9* el cuál dice que este dispositivo debe ser barato, pequeño y debe consumir poca energía. Está haciendo uso del sistema operativo Raspbian, una distribución Linux similar a Debian optimizada para los recursos disponibles, de este modo podremos conectarnos fácilmente en remoto para configurar el sistema y nos facilitará también la ejecución de los scripts necesarios.
- La siguiente pieza es, por tanto, la propia base de datos que almacenará toda esta información. Esta base de datos estará alojada en la nube a través de la plataforma Microsoft Azure [37], que haciendo uso del plan de estudiante es gratuita (con algunas limitaciones de tamaño y peticiones, pero para probar el correcto funcionamiento nos sirve). Además de almacenar las MACs capturadas también tendrá la información de los alumnos (relaciona el NIA con la MAC que está usando), de los días lectivos de cada asignatura y de las asistencias de cada alumno a cada asignatura.
- Una aplicación multiplataforma desarrollada mediante el framework Xamarin para ser compatible con Android, iOS y Windows. A través de ella tanto alumnos como profesores podrán registrarse, verificando que realmente lo sean, para que a continuación los profesores puedan activar y desactivar la toma de asistencias a cada una de sus clases, y los alumnos puedan confirmar su asistencia, así como ambos tipos de usuarios podrán consultar las asistencias almacenadas en el sistema.

## 4.3 Arquitectura de la parte de la Raspberry Pi

Esta parte es la encargada de escanear durante 30 segundos cada minuto en busca de nuevos dispositivos que se encuentren en el aula en la que se instaló la Raspberry Pi, e introducirlos en la base de datos para poder verificar la asistencia.
Para ello hemos desarrollado un script en Python que llama a otro script en bash para ejecutar la herramienta *airodump-ng* y recopilar las MACs presentes en ese momento, una vez las tiene, se conecta mediante la biblioteca *pyodbc* y el driver *tsql* a la base de datos alojada en el servidor de Microsoft Azure para introducir las nuevas (haciendo diferencia de conjuntos entre las anteriormente vistas y las recién recogidas). Además, este script hace uso de un fichero JSON a modo de fichero de configuración donde se define: el tiempo que se mantiene capturando, cada cuanto tiempo captura, el aula en el que está colocada la Raspberry Pi, así como el servidor, el nombre de usuario, contraseña, y el nombre de la base de datos.

## 4.4 Diseño de la base de datos

La base de datos empleada en este proyecto va a estar alojada en la nube, concretamente en un servidor de Microsoft Azure como se comentó anteriormente. El diseño de esta es el que se muestra en la *Figura 4-1*.
En él observamos cuatro tablas, cuyo objetivo es el siguiente:

- **temp_macs:** en ella se almacenan las asistencias que no han sido verificadas aún, bien sea porque la Raspberry Pi las ha detectado antes de que el alumno haya confirmado a través de la aplicación, o porque habiendo confirmado el usuario, la Raspberry Pi no hubiese detectado aún la MAC del alumno. En esta tabla almacenamos la MAC sujeto de la asistencia, el aula en la que tiene lugar la clase a confirmar asistencia y la fecha de inserción. Una vez se verifica en ambos extremos la asistencia, pasaría de forma definitiva a la tabla *asistencias*. Los tres campos forman la clave primaria por el motivo explicado en la siguiente tabla.
- **asistencias:** sirve para almacenar las asistencias una vez verificadas. En esta tabla tenemos el NIA del alumno, el aula en la que pasó asistencia, y la fecha de esta. Los tres campos forman la clave primaria, para permitir varias clases de un alumno en la misma aula, diferenciando por la fecha en que tiene lugar.
- **dias_lectivos:** sirve para contabilizar los días que ha habido clase de una misma asignatura, así como para permitir o no que los alumnos marquen su asistencia. Contiene el código de asignatura, la fecha en la que se habilitó y si aún se puede pasar asistencia. La clave primaria la forman los dos primeros campos.
- **user:** se usa para hacer corresponder una MAC con un alumno, identificado mediante su NIA. Por tanto, contiene el NIA y la MAC, formando ambos la clave primaria.

Se han definido índices auxiliares para hacer consultas más rápidas por alumno, MAC, aula o fecha.


**Figura 4-1: Modelo relacional de la base de datos usada por el sistema**

## *4.5 Arquitectura de la aplicación*

Lo primero que a uno le viene a la mente cuando tiene que desarrollar una aplicación con interfaz gráfica es el tradicional modelo de arquitectura MVC (Modelo Vista Controlador) [38]. En el caso de Android sería muy fácil de implementar, pues se disponen de los métodos y características necesarias (fragmentos, layouts, la clase View, las actividades, etc.) para llevarlo a cabo. Sin embargo, al utilizar Xamarin (cómo necesidad al *RNF 1*, el cuál dice que la aplicación debe ser multiplataforma), este modelo sufre variaciones y pasa a llamarse MVVM (Modelo Vista VistaModelo) [39]. La diferencia principal entre ambos modelos radica en que en este último la comunicación entre la vista y el modelo es en los dos sentidos a través del Modelo de Vista que es el que notifica los cambios a la Vista a través de *bindings*. El controlador muestra los datos en la vista y, si en la vista hay un cambio de datos, se actualiza el modelo automáticamente.

En nuestro proyecto no vamos a seguir al pie de la letra este modelo debido a la simplicidad de lo que tenemos que implementar, y a que lleva más tiempo y es más complicado de identificar todas las partes de cada ventana de la aplicación al tener que dividirlo en tres o cuatro archivos diferentes.
Las partes de las que va a disponer cada ventana son las siguientes:

- **Vista:** en ella se definen los elementos visibles por el usuario y su posicionamiento en la pantalla, pudiendo venir definidos los datos a visualizar desde el modelo. Esta se definirá en un archivo *.xaml* el cuál contendrá código XAML con la definición de la interfaz.
- **Modelo:** es la parte encargada de manipular los datos con los que se trabaja, tanto como recopilarlos y hacer algo con ellos, como para consultarlos y mostrarlos. Vendrá definido en un fichero con el mismo nombre que la vista, al que se le añade la extensión *.cs*, y en el cual se implementa la lógica que manipula tanto los datos como parte de la interfaz (ocultando o mostrando parte de los datos en función de otros datos) en el lenguaje C#.

La navegación entre ventanas tendrá como base un menú desplegable desde el lateral izquierdo de la aplicación, en el cual se visualizarán, pudiendo elegir con un toque en una de ellas, entre las distintas ventanas disponibles en cada momento. De forma adicional a esto, dentro de cada una de las ventanas se puede navegar a otra interaccionando mediante botones o al elegir un elemento de una lista. Esta nueva ventana se introduce en una pila de modo que se puede volver a la anterior ventana, al pulsar el botón retroceder propio del dispositivo, con solo eliminar la actual de la pila y recargar la anterior.

Además de las distintas ventanas de la aplicación se desarrollaron clases mediante las cuales acceder y recuperar distinta información de los ficheros JSON, otra para proporcionar la conexión a la base de datos, y otra para poder abrir los ajustes (que es dependiente de la plataforma en la que se ejecute la aplicación).

# 5 Desarrollo

## 5.1 Introducción

En este apartado se tratarán los aspectos más importantes del desarrollo de nuestro sistema. Tras las fases de definición del proyecto, análisis y diseño comentadas anteriormente, la siguiente fase ha sido la de codificación. Esta ha sido la fase del proyecto con mayor cantidad de investigación y aprendizaje por el hecho de realizarse con nuevas herramientas que no habían sido utilizadas anteriormente.
Para poder utilizar el framework Xamarin es necesario instalar varias herramientas en nuestro ordenador. Todas estas cuestiones se comentarán en el Manual del desarrollador en el Anexo B de este documento. A continuación, se comentará la estructura de las distintas partes del proyecto.

## 5.2 Desarrollo de la parte de la Raspberry Pi

La estructura de esta parte se explicó en el apartado 4.3 de esta memoria. Ahora vamos a explicar con detalle lo más importante del código desarrollado.
Empecemos por el script, ya que es la parte más grande.
En la *Figura C-1* podemos ver cómo leemos las distintas opciones del archivo de configuración. En el caso concreto del aula, se verifica que cumpla una expresión regular tal que el primer carácter de la cadena ha de ser una letra (L para indicar laboratorio, y A/B/C para indicar el edificio A, B y C, respectivamente) seguido de un guion, cuatro dígitos (indicando el número de aula) y una letra opcional (para contemplar casos como el laboratorio 5a y 5b). Un ejemplo de dicha cadena para el aula 1 del edificio A sería "A-0001", en el caso del laboratorio 5b sería "L-0005b".

En la *Figura C-2* se puede observar cómo nos conectamos a la base de datos, así como ejecutamos en un bucle el script encargado de capturar las MACs, para a continuación llamar a dos comandos que nos leen solo la parte que nos interesa (el primero es *sed*, con él leemos solo la última parte del fichero, desde donde aparece la cadena "Station MAC" que es a partir de donde salen los paquetes DHCP Discovery, y el segundo es *cut* que nos devuelve solo la primera columna que es justamente las MACs de los clientes), después de eso, introducimos en una lista todas las MACs quitando espacios en blanco y cabeceras, y por último, hacemos diferencia de conjuntos entre la lista anterior y todas las MACs recibidas anteriormente (almacenadas en la variable *macsVistas*).

Una vez disponemos de la lista de MACs, la recorremos una a una. Para cada una de ellas eliminaremos los puntos entre medias, la almacenaremos en la lista de vistas y comprobaremos si la MAC estaba en la base de datos sin validar, en ese caso la eliminaremos de la tabla temporal, obtendremos el NIA del alumno y la insertaremos como asistencia definitiva, en caso de que no estuviese, la insertará en la tabla temporal como se observa en la *Figura C-3*. Para hacer efectivos los cambios, se realizará un commit, ya que todas las operaciones se ejecutan automáticamente en una transacción, y se eliminará el archivo temporal, esperando el tiempo establecido en el archivo de configuración para volver a realizar la captura.

El script que ejecuta *airodump-ng* para leer las MACs cercanas se muestra en la *Figura C-4*.

El archivo de configuración JSON se muestra en la *Figura C-5*.

## 5.3 Desarrollo de la base de datos

La estructura de la base de datos se muestra en el apartado 4.4 de esta memoria. No hay nada más que merezca destacar más allá de que se usó DataGrip para la creación y visualización de tanto la base de datos como el modelo relacional.

## 5.4 Desarrollo de la aplicación

Como ya se ha comentado repetidas veces a lo largo de la memoria, hemos desarrollado una aplicación multiplataforma mediante el framework Xamarin. Este framework genera, dentro de una misma solución (así denominado a todo el proyecto, en nuestro caso la propia aplicación), un proyecto por cada plataforma además de un proyecto compartido.
Dentro del proyecto de cada plataforma se generan una serie de carpetas como las mostradas en la *Figura C-6* (a excepción de la carpeta *helpers* que es de creación propia y se explicará a continuación), con las diferencias en el nombre de los últimos archivos (que contiene un identificador de la plataforma), las librerías incluidas dentro de la carpeta *bin*, los archivos compilados dentro de *obj*, y el contenido de las carpetas *Assets* y *Resources* (que sirve para introducir imágenes, ficheros de texto y otro tipo de recursos).

La estructura del proyecto compartido se muestra en la *Figura C-7*, siendo todas las carpetas de creación propia para organizar mejor las cosas.
En la carpeta *clases* podemos encontrar diversas clases de utilidad para almacenar constantes de la aplicación (en el archivo *Constants.cs* mostrado en la *Figura C-8*), cargar y obtener datos de los ficheros JSON donde se almacenan los usuarios (alumnos, profesores), aulas y horarios (en el archivo *JSONLoader.cs* mostrado en la *Figura C-9*), y para realizar la conexión con la base de datos y almacenar las cadenas de las consultas SQL que se van a usar en distintos puntos de la aplicación (en el archivo *SQLExtensions.cs* mostrado en la *Figura C-11*).

En el fichero *clases*/*JSONLoader.cs* definimos métodos con los que poder abrir y leer los distintos ficheros JSON, así como extraer diversa información necesaria en distintas partes de la aplicación. En la *Figura C-9* se muestra el inicio de la clase, y el método que permite abrir y leer el archivo. Este método recibe como parámetro el nombre del fichero JSON a leer, que deberá estar ubicado en la carpeta *Recursos* que se ve en la *Figura C-7*, y dentro del mismo se crea la ruta de acceso a la carpeta en función de la plataforma que se ejecute en ese momento, se lee el fichero como Stream, y a continuación, se parsea mediante el método *JObject.Parse* de la librería *NewtonSoft JSON .NET*, devolviendo el resultado de este método que se trata de un objeto de tipo JObject.

En este mismo fichero se definen diversos métodos para extraer la información deseada, un ejemplo se puede ver en la *Figura C-10*. Todos los métodos de este tipo reciben como argumento el objeto devuelto por el método explicado arriba, usan sintaxis LINQ [40] (un tipo de sintaxis de C# similar a SQL que permite recorrer

colecciones y devolver un resultado) y devuelven los datos deseados. En el caso de ejemplo, primero se calcula el día de la semana comprobando que sea un día de diario (que no sea sábado o domingo), y a continuación, para cada aula del fichero, y para clase que hay en ese día, se busca si alguna de ellas empieza a la hora que se pasa por argumento, si la encuentra añade a una colección el identificador de la asignatura (esto lo hace automáticamente la sentencia *select*), y una vez se han recorrido todas, se devuelve esa colección (si no se encuentra ninguna, se devolverá una colección vacía). Se puede ver que LINQ es muy similar a SQL: dispone de *select*, condiciones mediante *where*, permite hacer "joins" (mediante varios comandos *from*) y además tiene otros comandos como *let*, el cual permite definir variables auxiliares para cada elemento de la colección para poder almacenar un cálculo intermedio.

En el fichero *clases/SQLExtensions.cs* se declara un método que permite conectarse a la base de datos de forma más cómoda, así como todas las sentencias SQL que usa la aplicación. En la *Figura C-11* se pueden observar algunas de ellas.

El método de conexión se puede ver en la *Figura C-12*. En él se pueden pasar como argumentos el nombre de usuario y la contraseña de la base de datos, así como en servidor y el nombre de esta, en caso de no pasarlos se usarán los definidos en el archivo de constantes.
Dentro del método creamos una nueva conexión haciendo uso de la clase auxiliar *SqlConnectionStringBuilder*, la cual nos ayuda a identificar y pasar los argumentos a la conexión estableciendo su propiedad *ConnectionString* al homónimo del *builder*, para finalmente, terminar devolviendo la conexión.

Una vez vista la carpeta *clases*, podemos continuar con *interfaces*, en esta carpeta se encontraría un único archivo llamado *IOpenWifiSettings.cs* el cual tiene el cometido de indicar los métodos necesarios a implementar por clases que implementen la interfaz para poder abrir los ajustes Wifi. Esto se hace así debido a que cada plataforma tiene una forma distinta de abrir los ajustes, por tanto, lo que se hace es declarar una interfaz común, y, a través de *Dependency Injection*, introducir en el momento de compilación de cada uno de los proyectos el método correcto. Esta interfaz se muestra en la *Figura C-13*.

En Android, para abrir los ajustes Wifi se hace de la forma que se muestra en la *Figura C-14*.
En el caso de iOS, se hace tal y como se observa en la *Figura C-15*.
Por último, para Windows Phone se hace como se muestra en la *Figura C-16*.
La siguiente carpeta del proyecto compartido es *Recursos*, esta carpeta, como ya se dijo antes, contiene los archivos JSON de los que hará uso la aplicación. En este caso son cuatro: *asignaturas.json*, *aulas.json*, *matriculas.json* y *users.json*.

El fichero *users.json* está estructurado como muestra la *Figura C-17*.
En el caso del fichero *matriculas.json*, se puede ver en la *Figura C-18*.
Para el fichero *asignaturas.json* tenemos una estructura como la de la *Figura C-1*9.
Y, por último, el fichero *aulas.json* es como muestra la *Figura C-20*. Hay que destacar que dentro de cada aula tenemos una lista de días (de lunes a viernes) y donde en cada día hay una lista de clases que tienen lugar, el cual actúa como horario.

Por último, ya solo queda la carpeta *VISTAS* en la cual se hayan tanto el código XAML para el diseño de la interfaz como el *Code Behind* correspondiente a cada una de ellas en el que se implementa la funcionalidad.

El punto de entrada de la aplicación a través del cual nos podemos desplazar entre las distintas ventanas de la aplicación se encuentra en el fichero *Principal.cs* y cuyo aspecto se muestra en la *Figura C-21*.
Se trata de un *MasterDetailPage*, un tipo de página en el cuál tenemos un menú desplegable desde la izquierda donde se pueden elegir las distintas ventanas disponibles y navegar entre ellas (el *Master*), y que muestra la ventana actual en el centro (el *Detail*). Cuando carga la aplicación se actualiza tanto la lista de ventanas disponibles como la página de detalle en función de si habías iniciado sesión o no, en caso afirmativo, comprueba si el usuario es profesor o alumno para mostrar como ventana principal la ventana donde introducir las asistencias o la ventana donde se pueden visualizar las asistencias, en otro caso se mostraría la ventana de registro.
Al seleccionar un elemento de la lista de *Master* se crea una instancia de la página elegida y se navega a ella sustituyendo *Detail* por una *NavigationPage* de ella.

Vamos a poner como ejemplo la ventana de confirmar asistencias, que se encuentra en el fichero *ConfirmAttendancePage.xaml* y cuyo código se puede observar en la *Figura C-22.*
Se trata de un *layout* que contiene a su vez dos *layouts* distintos para mostrar uno u otro, mediante el *Code Behind*, dependiendo de si se permite marcar o no la asistencia. En caso de permitirse, se mostrará el primero de ellos. Se trata de una cuadricula de dos filas y una única columna, donde la primera fila contiene tres etiquetas de texto que permiten generar un texto que indica la asignatura y el aula en la que se encuentra el usuario y para la cual va a pasar asistencia el alumno (o habilitarla/deshabilitarla en el caso del profesor), a las cuales se les asigna un nombre identificativo mediante *x:Name* para posteriormente poder acceder a ellas; y la segunda fila contiene el botón que permite confirmar o habilitar/deshabilitar la asistencia. En el caso de que no se pueda realizar la acción por algún motivo, se mostrará el segundo layout, que simplemente contiene una etiqueta, con el texto en color rojo, identificada con un nombre para introducir el mensaje informativo.
Con ello hemos explicado cómo se diseña la interfaz de una ventana, veamos ahora cómo funciona el *Code Behind* de esa ventana, el cual se muestra en las figuras *C-23* a *C-25*.

El constructor, el cual se invoca en el momento de cambiar de página, inicializa toda la interfaz, y una vez se va a mostrar al usuario se llama automáticamente al método *OnAppearing*, donde llamamos de forma asíncrona, mediante la palabra reservada *await* que necesita declarar el método como *async* para indicar que es asíncrono, a nuestro método *UpdateUI* para actualizar la interfaz sin bloquear la aplicación.
Este método deberá ser asíncrono también ya que en algún momento se llama a otro método de forma asíncrona. Lo primero que se hace es sacar de las preferencias si el usuario es profesor o alumno para poner el título correcto a la ventana, así como obtener y almacenar la fecha actual para usarla posteriormente, esto se puede ver en la *Figura C-23.* A continuación, se lee el archivo JSON de usuarios, para en función de si es alumno o profesor, escribir el texto correcto en los componentes de la interfaz y recuperar la lista de asignaturas del usuario, como se puede observar en la *Figura C-24*. Después se lee el archivo de JSON de las aulas para obtener la lista de asignaturas que empiezan ahora y combinar con la lista de asignaturas del alumno,

obteniendo la primera de ellas que se encuentre en ambas listas. En caso de no haber ninguna se mostraría el segundo *layout* con el mensaje de error, y si hubiese continuamos comprobando que la asistencia para esa clase estuviese abierta. Si no lo está se mostraría el mensaje de error, y si lo estuviese, cargaríamos el JSON de asignaturas para obtener el nombre y el aula (esto mediante el JSON de aulas abierto anteriormente), y poder completar los elementos restantes de la interfaz, habilitando el primer *layout*. Esto ultimo se puede ver en la *Figura C-25*.

Para comprobar si la asistencia está activada nos conectamos a la base de datos mediante la clase auxiliar *SQLExtensions* y ejecutamos el *select* correspondiente, en caso de que se devuelva información leemos un valor booleano que nos indica cuando vale t*rue* que sí está habilitada, o *false* cuando no lo está porque el profesor había cerrado la asistencia, y en caso de que no hubiese nada es porque el profesor aún no lo ha habilitado, por tanto, devolvemos *false*. Esto se hace en el método *IsAttendanceOpen* el cuál devuelve un objeto de tipo *Task<bool>* debido a que será ejecutado de forma asíncrona, haciendo *await* en su llamada, y se indica mediante el *Task*. Se puede ver en la *Figura C-26*.

Cuando se pulsa el botón para confirmar o habilitar/deshabilitar la asistencia se comprueba también si es profesor o alumno, en el caso del alumno comprobamos que pueda marcar la asistencia con el método anterior, si no puede se informa de ello mediante un cuadro de dialogo, y además debemos verificar que no hubiese pulsado para la misma asignatura el botón anteriormente, si es así se le informaría del mismo modo; y en el caso del profesor, si no había pulsado el botón se añadiría en base de datos un registro que permita tomar asistencias mediante el método *AcceptAttendanceAsync*, y en caso contrario se actualizaría este registro para deshabilitarlo mediante el método *RejectAttendanceAsync*. Esto se muestra en la *Figura C-27*.
Ambos métodos simplemente se conectan a la base de datos e insertan o actualizan un campo, en función de la asignatura y la fecha actual.

En el caso del resto de ventanas sería bastante similar la forma de programar tanto la vista como el *Code Behind*, por tanto, no se explican de forma extensiva, tan solo vamos a destacar lo más importante.

Cuando el usuario abre por primera vez la aplicación le aparece una ventana de registro en la cual se debe elegir mediante un *switch* si eres un estudiante o un profesor, en caso de ser profesor simplemente deberás introducir el correo electrónico para poder verificar que eres un profesor de la UAM, y en el caso del alumno, además deberá introducir su MAC, para ello la aplicación le abrirá los ajustes Wifi donde podrá localizar y recordar su MAC para introducirla en la aplicación (esto se hizo así ya que iOS no permitía leerlo directamente a través del código), de igual forma se verificará que sea un estudiante de la UAM y se le registrará en la base de datos como usuario de la aplicación. Por último, se guardará en las preferencias de la aplicación (mecanismo de persistencia aportado por el framework) si el usuario es alumno o profesor, así como el email en caso del profesor, y en el caso del alumno, el NIA (obtenido del fichero JSON de alumnos) y la MAC, para poder usarlos a lo largo de toda la interacción del usuario en la aplicación. Todo lo que se explicó tiene lugar en los archivos *RegisterPage.xaml* y *RegisterPaga.xaml.cs*, donde se encuentran la vista y el comportamiento de forma respectiva.

En el caso de la ventana que permite visualizar las asistencias (*CheckAttendancePage.xaml* y *CheckAttendancePage.xaml.cs*) la interfaz se compone únicamente de una lista en la cual se muestran todas las asignaturas del usuario, y al pulsar sobre una de ellas se navega, mediante una pila haciendo uso del método *Navigation.PushAsync*, a otra ventana denominada *AttendanceList* (tanto *.xaml* como *.xaml.cs*) en la cual se sitúa otra lista que contiene para cada alumno de esa asignatura (o solo una fila en caso de ser el estudiante el que pulsó sobre el elemento) el porcentaje de faltas a esa asignatura. Para ello solo necesitamos contar los días que hubo clase de esa asignatura, que los tenemos almacenados en la tabla *días_lectivos*, restarle a esos días los días en los que el alumno verificó su asistencia, y dividir esa resta entre el total de días de la asignatura.

Por último, solo nos falta la página de ajustes (*SettingsPage.xaml* y *SettingsPage.xaml.cs*), en la cual de momento solo podremos ver la información del usuario que se encuentra logueado en la aplicación y cerrar sesión para poder entrar con un usuario distinto.

En el Anexo D se muestran las capturas de pantalla de la aplicación, realizadas en un dispositivo físico.

# 6 Integración, pruebas y resultados

## *6.1 Introducción*

La fase de pruebas se ha realizado para encontrar errores cometidos durante el desarrollo, posibles mejoras de usabilidad o la inclusión de nuevas funcionalidades. Como se ha comentado anteriormente, se estableció un desarrollo ágil entre la fase de desarrollo y la fase de pruebas, así que al finalizar un periodo de pruebas se volvía a codificar para cambiar todos aquellos aspectos susceptibles de mejora. En este capítulo se realiza una descripción de las pruebas realizadas y los resultados obtenidos.

## *6.2 Pruebas*

Para realizar las pruebas se han utilizado los emuladores y dispositivos físicos comentados en el apartado 3.4.1 de esta memoria. La ventaja de realizar las pruebas en emuladores es que las puedes realizar en el mismo equipo en el que se está codificando, siendo un proceso muy rápido el de modificar el código, pudiendo hacerlo a la vez que haces las pruebas y visualizando la interfaz tal y como se mostraría en un dispositivo real.
Dentro de las pruebas realizadas podemos diferenciar tres tipos: las pruebas unitarias en las que se probaba cada ventana por separado, y el script de forma independiente; las pruebas de compatibilidad en las que se verifica que la aplicación sea compatible con distintas plataformas, así como tamaños y resoluciones de pantalla; y las pruebas de validación y verificación donde se prueba el *prototipo* o el producto final en su ámbito de uso.

### 6.2.1 Pruebas unitarias

En este tipo de pruebas se evaluaron cada una de las ventanas de las que dispone la aplicación por separado para comprobar que su funcionamiento era el correcto, mostrando los datos o información correcta en todas las situaciones posibles, así como verificar que el script de la Raspberry Pi funcione correctamente introduciendo las MACs en la base de datos.
De este modo se podía asegurar que cada uno de los requisitos fuese satisfecho.

### 6.2.2 Pruebas de compatibilidad

Al haber realizado una aplicación multiplataforma, en este tipo de pruebas se intentó comprobar cómo funciona la aplicación con diferentes sistemas operativos. Al disponer únicamente de dispositivos físicos Android, las pruebas correspondientes a otros sistemas operativos se tuvieron que suspender ya que no hay manera de usar un emulador iOS sin disponer de un ordenador Mac donde compilar la aplicación.
También hemos hecho pruebas de compatibilidad con diferentes tamaños de pantalla. Para ello hemos utilizado los dispositivos físicos Android comentados en el apartado 3.4.1 y comprobamos que la aplicación se mostraba correctamente en ambos.

### 6.2.3 Pruebas de validación y verificación

Este tipo de pruebas se realizaron en distintos momentos del día simulando distintas clases con distinto número de alumnos en cada una con el principal objetivo de evaluar el correcto funcionamiento y la usabilidad del sistema. Al haber realizado anteriormente pruebas unitarias, el número de fallos detectados en esta fase era mínimo. Para realizarlas se instaló el sistema tal y cómo se explica en el Anexo A de este documento.

## *6.3 Resultados*

El resultado final de este Trabajo de Fin de Grado es un sistema software que permite sustituir las anotaciones manuales de asistencia de los alumnos a las diversas clases de la UAM. Tras diversas pruebas y prototipos se llegó a una versión final que puede ser puesta en funcionamiento en las aulas.
Los resultados de estas pruebas fueron satisfactorios ya que el sistema cumplió todos los requisitos, tantos funcionales como no funcionales. Tras haber superado con éxito dichas pruebas se considera finalizado el desarrollo del sistema y listo para ponerlo en marcha y utilizarlo en clases reales.

# 7 Conclusiones y trabajo futuro

## 7.1 Conclusiones

Tras la realización de este Trabajo de Fin de Grado se ha conseguido desarrollar un sistema que sustituye las anotaciones manuales en un aula sobre las asistencias de los alumnos. Además, se ha utilizado por primera vez el framework Xamarin para construir aplicaciones móviles multiplataforma.

Se considera que la realización del proyecto ha sido satisfactoria ya que se han cumplido los objetivos planteados inicialmente. La mayor dificultad a la hora de llevar a cabo su realización ha sido familiarizarse con el nuevo entorno, la nueva arquitectura MVVM y la estructura novedosa del proyecto con la que no se había trabajado nunca anteriormente. Además del éxito en el cumplimiento de los objetivos planteados también es satisfactorio el hecho de haber aprendido a utilizar una nueva herramienta que parece bastante interesante.

Durante la realización de este Trabajo de Fin de Grado se han aplicado los conocimientos aprendidos en varias asignaturas del Grado en Ingeniería Informática. En concreto se han utilizado los conocimientos sobre diseño de bases de datos aprendidos en *Estructura de Datos* (modelado y manejo de bases de datos); los conocimientos relacionados con el análisis y diseño de proyectos y la educción de requisitos aprendidos en las asignaturas de *Proyecto de Análisis y Diseño de Software*, *Ingeniería del Software* y *Proyecto de Ingeniería del Software*; y los conocimientos sobre el diseño y el flujo de una aplicación móvil aprendidos en la asignatura de *Desarrollo de Aplicaciones para Dispositivos Móviles.* El haber podido emplear los conocimientos adquiridos a lo largo de la carrera ha hecho que la experiencia de la elaboración de este Trabajo de Fin de Grado haya sido aún más satisfactoria y ha permitido afianzar y reforzar dichos conocimientos.

En el siguiente subapartado se expondrán posibles mejoras a realizar sobre el sistema, ya que en esta primera versión se ha buscado lograr la funcionalidad deseada y no se ha hecho tanto énfasis en algunos aspectos como el diseño gráfico de la aplicación o funcionalidades extras no imprescindibles. Además, se espera que con la utilización del sistema por parte de varios usuarios se vayan recibiendo sugerencias de mejora o de nuevas funcionalidades a añadir.

## 7.2 Trabajo futuro

Los objetivos que se pretendían alcanzar con este proyecto se pueden considerar cumplidos con éxito, no obstante, todavía hay ciertas cosas susceptibles de mejora en un trabajo futuro como son:

- **Hacer más genérica la aplicación de modo que pueda ser usada en otras universidades o centros escolares:** tal y cómo hemos desarrollado la aplicación solo admite usuarios de la UAM, ya que se verifica una estructura concreta de dirección de correo electrónico y de nombre de las aulas. Si el sistema se quisiese extender, debería hacerse de forma genérica esta parte, tal vez añadiendo un archivo o una tabla en base de datos que relacione alumnos con universidades.

- **Permitir elegir el idioma de la aplicación:** en un principio se desarrolló en un único lenguaje, el castellano, por rapidez de uso, pero es previsible que alguien prefiera usarla en inglés.
- **Mejorar el diseño de la interfaz de la aplicación:** se podría pulir el diseño de la aplicación para hacerla más atractiva, así como crear y añadir un logo identificativo de la misma.
- **Añadir notificaciones a los alumnos de cada asignatura, para que cuando el profesor habilite la asistencia los alumnos sean notificados y puedan confirmar su asistencia rápidamente:** no se ha investigado la forma de llevar esto a cabo, pero sería una característica muy buena que podría diferenciar a este sistema de otro similar que fuese implementado.
- **Posibilidad de añadir o eliminar tanto alumnos como asignaturas por parte del profesor, así como gestionar el horario de cada una de ellas:** de este modo la gestión sería más rápida y no conllevaría una actualización de la aplicación, para ello toda la información que actualmente se encuentra en archivos JSON debería trasladarse a una base de datos.
- **Posibilidad de exportar o enviar por correo un fichero con las faltas de todos los alumnos de una clase:** de este modo el profesor podría manejar estos datos de la forma a la que seguramente está acostumbrado.
- **Controlar mejor los engaños:** actualmente cualquiera puede registrarse en la aplicación haciéndose pasar por otro alumno o profesor, ya que no se pide ninguna información secreta conocida solo por cada usuario.

# Referencias


[1] *Xamarin Framework*:
https://docs.microsoft.com/es-mx/xamarin/ (consultado por última vez el 03/06/2018)
[2] *Google Maps:*
https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=es
(consultado por última vez el 03/06/2018)
[3] *NFC:*
https://es.wikipedia.org/wiki/Near_field_communication (visitado por última vez el 03/06/2018)
[4] *Google Pay:*
https://pay.google.com/intl/es_es/about/ (visitado por última vez el 03/06/2018)
[5] *Bizum:*
https://bizum.es/ (consultado por última vez el 05/06/2018)
[6] *Google Play:*
https://es.wikipedia.org/wiki/Google_Play (consultado por última vez el 05/06/2018)
[7] *App Store:*
https://es.wikipedia.org/wiki/App_Store (consultado por última vez el 05/06/2018)
[8] *Número de aplicaciones en cada tienda:*
https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ (consultado por última vez el 05/06/2018)
[9] *Duolingo:*
https://play.google.com/store/apps/details?id=com.duolingo (consultado por última vez el 05/06/2018)
[10] *Agenda escolar:*
https://play.google.com/store/apps/details?id=daldev.android.gradehelper (consultado por última vez el 05/06/2018)
[11] *Udemy:*
https://play.google.com/store/apps/details?id=com.udemy.android (consultado por última vez el 05/06/2018)
[12] *Kahoot:*
https://play.google.com/store/apps/details?id=no.mobitroll.kahoot.android (consultado por última vez el 05/06/2018)
[13] *LiveBoard:*
*https://play.google.com/store/apps/details?id=com.inconceptlabs.liveboard* (consultado por última vez el 05/06/2018)
[14] *Attendance:*
https://play.google.com/store/apps/details?id=com.petermanapps.atcloud&rdid=com.petermanapps.atcloud (consultado por última vez el 06/06/2018)
[15] *Attendance Register (students):*
https://play.google.com/store/apps/details?id=com.rudra.attendanceRegister (consultado por última vez el 06/06/2018)
[16] *Pocket Attendance:*
https://play.google.com/store/apps/details?id=com.rnl.pocketattendance&rdid=com.rnl.pocketattendance (consultado por última vez el 06/06/2018)
[17] *Smart Attendance:*
https://play.google.com/store/apps/details?id=com.alienartsoftwares.smartattendance.free
(consultado por última vez el 06/06/2018)

[18] *Control de asistencia a clase Universidad Pontificia de Salamanca:*
http://www.dicyt.com/noticias/control-de-asistencia-a-clase-mediante-el-movil
(consultado por última vez el 09/06/2018)
[19] *Desarrollo ágil:*
https://www.renacen.com/blog/principios-del-desarrollo-agil-metodologias-agiles/
(consultado por última vez el 09/06/2018)
[20] *Raspberry Pi:*
https://www.raspberrypi.org/ (consultado por última vez el 10/06/2018)
[21] *Android:*
https://www.android.com/intl/es_es/ (consultado por última vez el 10/06/2018)
[22] *iOS:*
https://support.apple.com/es-es/ios (consultado por última vez el 10/06/2018)
[23] *Windows Phone:*
http://windowsphone.com/ (consultado por última vez el 10/06/2018)
[24] *Python:*
https://www.python.org/ (consultado por última vez el 10/06/2018)
[25] *Bash scripting:*
https://es.wikipedia.org/wiki/Bash (consultado por última vez el 10/06/2018)
[26] *Microsoft SQL Server:*
https://es.wikipedia.org/wiki/Microsoft_SQL_Server (consultado por última vez el 10/06/2018)
[27] *DHCP Discovery:*
https://es.wikipedia.org/wiki/Protocolo_de_configuraci%C3%B3n_din%C3%A1mica_de_host#DHCP_Discovery (consultado por última vez el 10/06/2018)
[28] *C#:*
https://es.wikipedia.org/wiki/C_Sharp (consultado por última vez el 11/06/2018)
[29] *XAML:*
https://es.wikipedia.org/wiki/XAML (consultado por última vez el 11/06/2018)
[30] *JSON:*
https://es.wikipedia.org/wiki/JSON (consultado por última vez el 11/06/2018)
[31] *Visual Studio:*
https://es.wikipedia.org/wiki/Microsoft_Visual_Studio (consultado por última vez el 11/06/2018)
[32] *Atom:*
https://es.wikipedia.org/wiki/Atom_(software) (consultado por última vez el 11/06/2018)
[33] *DataGrip:*
https://www.jetbrains.com/datagrip/ (consultado por última vez el 11/06/2018)
[34] *Git:*
https://es.wikipedia.org/wiki/Git (consultado por última vez el 11/06/2018)
[35] *Bitbucket:*
https://es.wikipedia.org/wiki/Bitbucket (consultado por última vez el 11/06/2018)
[36] *NewtonSoft JSON:*
https://www.newtonsoft.com/json (consultado por última vez el 11/06/2018)
[37] *Microsoft Azure:*
https://es.wikipedia.org/wiki/Microsoft_Azure (consultado por última vez el 12/06/2018)
[38] *MVC:*
https://en.wikipedia.org/wiki/Model-view-controller (consultado por última vez el 12/06/2018)

[39] *MVVM:*
https://docs.microsoft.com/es-es/xamarin/xamarin-forms/enterprise-application-patterns/mvvm (consultado por última vez el 12/06/2018)
[40] *LINQ:*
https://es.wikipedia.org/wiki/Language_Integrated_Query (consultado por última vez el 13/06/2018)
[41] *Instalación de Raspbian:*
https://raspberryparatorpes.net/instalacion/noobs-paso-a-paso-instalar-el-sistema-operativo-en-la-raspberry-pi/ (consultado por última vez el 19/06/2018)
[42] *Pasos para instalar el driver de red:*
https://github.com/seemoo-lab/nexmon#build-patches-for-bcm43430a1-on-the-rpi3zero-w-using-raspbian-stretch-recommended (consultado por última vez el 19/06/2018)
[43] *Script automático para la instalación del driver:*
https://gist.github.com/someguycrafting/ce830dd957e1098fcaa051e42fc18db1 (consultado por última vez el 19/06/2018)
[44] *Instalación de Xamarin en Wndows:*
https://docs.microsoft.com/es-mx/xamarin/cross-platform/get-started/installation/windows (consultado por última vez el 19/06/2018)
[45] *Descarga de Visual Studio:*
https://visualstudio.microsoft.com/vs/ (consultado por última vez el 19/06/2018)

# Glosario

| | |
|---|---|
| Alert | Ventana emergente que aparece en las aplicaciones para mostrarle avisos al usuario relacionados con su actividad o el estado de la aplicación. |
| Colección | Tipo de dato similar a una lista, pero más general ya que no garantiza ni orden ni ninguna operación concreta más que su iteración o recorrido. |
| Framework | Representa una arquitectura de software con un conjunto de herramientas y métodos que sirven como referencia para realizar un desarrollo posterior. |
| JSON | Acrónimo de JavaScript Object Notation, se trata de un lenguaje de marcado ligero utilizado para la transmisión y el almacenaje de datos. |
| MVC | Siglas de Modelo-Vista-Controlador, un patrón de arquitectura de software. |
| MVVM | Siglas de Modelo-Cista-VistaModelo, un patrón de arquitectura software similar a MVC. |
| Code Behind | Parte del código que especifica el comportamiento de una vista en Xamarin. |
| Prototipo | Producto que se obtiene al finalizar una iteración de ciclo de vida de un desarrollo ágil y que puede ser utilizado en el entorno de destino pese a no tener la funcionalidad completa. |
| Smartphone | Tipo de teléfono móvil inteligente que permite realizar numerosas actividades y que puede tener un comportamiento similar a una computadora. |
| Tablet | Dispositivo electrónico que tiene un tamaño intermedio entre un ordenador y un teléfono móvil. |
| .apk | Extensión de archivos correspondientes a aplicaciones Android, su nombre completo es Android Application Package, pero se suele llamar apk. |
| .ipa | Extensión de los archivos correspondientes a aplicaciones para dispositivos iOS. |

# Anexos

## A Manual de instalación

En este anexo se van a explicar los pasos necesarios para poner el sistema al completo en funcionamiento. Para ello se va a dividir en dos pasos: el primero en el cual alguien instalará y configurará la Raspberry Pi de cada aula; y el segundo en el que tanto alumnos como profesores se deberán descargar e instalar la aplicación en sus dispositivos móviles.

Antes de empezar a montar el sistema hay una serie de requisitos previos:

- Disponer de una base de datos SQL Server estructurada de la forma que se muestra en la *Figura 4-1*.
- Haber introducido correctamente tanto las clases como los horarios de estas, así como la información de alumnos y profesores incluyendo las clases a las que asiste cada uno, en los correspondientes archivos JSON de la aplicación móvil destinados para tal fin.
- Disponer de conexión a Internet por cable en cada una de las aulas en las que se vaya a instalar una Raspberry Pi, para poder hacer consultas a la base de datos mencionada anteriormente.

Para configurar la Raspberry Pi de cada aula se seguirán los siguientes pasos:

1) Antes de poder llevar a cabo la instalación se debe disponer de una Raspberry Pi, su correspondiente adaptador de corriente, y una tarjeta microSD con el sistema operativo Raspbian instalado. Se asume que se sabe hacer esto, si no es el caso se puede consultar la referencia número 41.
2) Para poder capturar los paquetes DHCP Discovery que envían los diversos clientes de alrededor, debemos poder poner la tarjeta de red en modo monitor. En nuestro caso queríamos usar la integrada en la Raspberry Pi 3 y para ello necesitamos cambiar el driver de esta, pero se podría usar una tarjeta Wifi externa en caso de disponer de un modelo diferente, ya que el driver que mencionamos únicamente es compatible con el modelo 3 y el Zero W. Seguimos los pasos indicados en la referencia 42, y también se podría hacer uso del script de la referencia 43, aunque preferimos la primera opción.
    a. Una vez listo, reiniciamos para que arranque el driver correcto.
    b. A continuación, y tras cada reinicio del sistema, debemos ejecutar dos comandos que activan el modo monitor en la tarjeta, estos son:
    *sudo iw dev wlan0 interface add mon0 type monitor* y
    *sudo ifconfig mon0 up*
3) Instalamos la suite *aircrack-ng*, la cual nos permitirá usar mediante nuestro script la herramienta *airodump-ng*, para ello basta con ejecutar en una terminal *sudo apt intall aircrack-ng*.
4) Creamos el archivo de configuración como corresponda para el entorno en el que se despliegue el sistema. Este es un JSON como el mostrado en la *Figura C-4*.
5) Con estos pasos terminados, solo nos faltaría ejecutar nuestro script Python desde la misma carpeta en la que se creo el archivo de configuración para que empiece a capturar las MACs cercanas.

Para la instalación de la aplicación los usuarios idealmente accederían a la tienda de aplicaciones correspondiente a su sistema operativo y descargarían e instalarían la aplicación, pero actualmente no se ha distribuido en estas tiendas y, por tanto, se deberían suministrar de algún otro modo a los usuarios los paquetes de instalación para cada plataforma. Una vez hecho, cada uno se podrá instalar la aplicación siguiendo una serie de pasos diferentes en cada plataforma.

En el caso de Android estos pasos pasan por localizar el archivo *.apk* de la aplicación recién descargado y hacer click sobre él para instalarlo, puede ocurrir que no se tengan activados los "orígenes desconocidos" en ese caso una ventana emergente te explica como activarlos y una vez activados al volver a hacer click podrás continuar, abriéndose un dialogo que te pide confirmar la instalación, para finalmente mostrar que se instaló correctamente.

En otra plataforma serán pasos similares, posiblemente con otro tipo de archivo, como por ejemplo en iOS un archivo *.ipa*.

Una vez instalada la aplicación los usuarios podrán registrarse y empezar a introducir y visualizar asistencias.

# B Manual del programador

En este apartado de la memoria se va a comentar cómo instalar las herramientas necesarias para este proyecto, en concreto nos centraremos en la instalación y uso del framework Xamarin.

Para el desarrollo de la base de datos y los scripts que se van a ejecutar en la Raspberry Pi necesitaremos el editor de texto Atom (consultar [32], u otro similar que permita abrir ficheros JSON, Python y Bash), así como el gestor de bases de datos Datagrip (consultar [33], u otro similar que permita usar SQL Server).
Como ya se comentó en la guía de instalación del *Anexo A* deberemos instalar un driver que nos permita poner la tarjeta inalámbrica en modo monitor, o usar una externa, además de instalar la *suite* de herramientas *aircrack-ng*.

Para el desarrollo de la aplicación móvil necesitaremos instalar Visual Studio junto al plugin de Xamarin, un tutorial detallado de cómo hacerlo en Windows se puede ver en [44].
El primer paso es descargar Visual Studio de su página oficial [45] y ejecutar el instalador, una vez abierto debemos elegir la opción "*Desarrollo para dispositivos móviles con .NET"* para, a continuación, finalizar la instalación.
Tras haber instalado tanto Visual Studio como Xamarin ya podemos crear una aplicación móvil multiplaforma. Para ello debemos crear un nuevo proyecto en Visual Studio eligiendo en el apartado "*Visual C#"* la opción "*Cross Platform"*, darle un nombre al proyecto y pulsar en continuar, elegir en la siguiente ventana "*Aplicación en blanco"* y como "*Proyecto compartido"* en la parte "*Estrategia de uso compartido de código"*, así como seleccionar las plataformas en las que se va a poder ejecutar la aplicación, para pulsar *OK* y poder empezar el desarrollo.
Para crear nuevas páginas tan solo debemos pulsar click derecho sobre el proyecto compartido y elegir la opción "*Agregar" -> "Nuevo elemento"* donde pulsaremos sobre la opción *"Página en blanco" y le daremos el nombre deseado a la misma.*
Estas opciones generan la estructura de carpetas y los archivos necesarios, los cuales deberemos modificar a nuestra necesidad para implementar las funcionalidades deseadas.
Para probar una aplicación tan solo bastará con elegir en la parte superior de Visual Studio la plataforma de destino, si deseamos lanzar la aplicación en modo de depuración o no, y el destino donde se instalará la misma, y finalmente para poder ejecutarla debemos pulsar el botón *play*. En caso de querer ejecutarlo en un emulador deberemos haber creado uno mediante el "*Android Device Manager"*, y en el caso de un dispositivo físico deberemos tenerlo conectado al mismo ordenador donde estemos programando mediante USB y con las opciones de depuración habilitadas (en el caso de Android).
Para generar los diversos paquetes de la aplicación para cada sistema operativo deberemos hacer click derecho uno a uno sobre cada uno de los proyectos de dichas plataformas y pulsar sobre *Archivar*, de este modo se generarán los archivos necesarios en las respectivas carpetas de cada uno de estos proyectos.

## C Capturas del código

```python
import syslog
import pyodbc
import subprocess
import time
import datetime
import json
import re

LOG_PREFIX = "[MAC_SNIFFER_"

syslog.openlog(logoption=syslog.LOG_PID)

with open('config.json') as config:
    data = json.load(config)

    # maximo 10 caracteres (L delante para labs)
    AULA = data['aula']

    LOG_PREFIX += AULA + "] "

    if len(aula) > 10:
        errMessage = "La longitud de la cadena 'aula' debe ser " \
                     + "menor o igual a 10 caracteres."
        syslog.syslog(LOG_ERROR, errMessage)
        raise Exception(errMessage)
    if not re.match(r"[LABC]-[0-9]{4}[a-z]*", aula):
        errMessage = "La cadena 'aula' no tiene el formato correcto."
        syslog.syslog(LOG_ERROR, errMessage)
        raise Exception(errMessage)

    update_interval = data['capture_update_interval']
    capture_duration = data['capture_duration']
    capture_interval = data['capture_interval']
    server = data['server']
    database = data['database']
    username = data['username']
    password = data['password']
```

**Figura C-1: Lectura del archivo de configuración en el script Python**

```
macsVistas = []

try:
    cnxn = pyodbc.connect(
        'DRIVER={freetds};SERVER=' + server + ';PORT=1433;DATABASE=' + database
        + ';UID=' + username + ';PWD=' + password + ';TDS_VERSION=8.0')

    hora = datetime.datetime.now().strftime("%d-%m-%Y_%H-%M-%S")

    syslog.syslog(LOG_INFO, LOG_PREFIX + 'conexion exitosa ' + hora)

    cursor = cnxn.cursor()

    while True:

        hora = datetime.datetime.now().strftime("%d-%m-%Y_%H-%M-%S")
        filename = hora + '.csv'

        subprocess.call(['./capturaMACS.sh', hora, str(update_interval),
                         str(capture_duration)])

        # devuelve desde Station MAC hasta el final del archivo
        sedCmd = ['sed', '-n', '/Station MAC/,$ p', filename]
        sed = subprocess.Popen(sedCmd, stdout=subprocess.PIPE)

        # devuelve solo la primera columna
        cutCmd = ['cut', '-d', ",", '-f', '1']
        cut = subprocess.Popen(cutCmd, stdin=sed.stdout, stdout=subprocess.PIPE)

        lista = map(lambda x: x.strip(), list(cut.stdout))[1:-1]

        # las que estan en la lista pero no en las vistas
        lista = list(set(lista) - set(macsVistas))

        syslog.syslog(LOG_INFO, "nuevas macs = " + lista)

        for mac in lista:
            mac = str.replace(mac, ':', '')
            macsVistas.append(mac)

            cuenta = cursor.execute(count_macs_STMT, mac).fetchval()
```

**Figura C-1: Conexión a base de datos y obtención de MACs en el script Python**

```
            if cuenta and cuenta > 0:
                syslog.syslog(LOG_INFO, "mac estaba en BD sin validar = " + mac)

                cursor.execute(delete_macs_STMT, mac)

                syslog.syslog(LOG_INFO,
                              "eliminadas todas las entradas de hoy con esa mac")

                row = cursor.execute(get_nia_STMT, mac).fetchone()
                if row:
                    syslog.syslog(LOG_INFO, "alumno con nia = " + row.nia
                            + " presente en el aula")

                    cursor.execute(insertAsistSTMT, row.nia)

                    syslog.syslog(LOG_INFO, "validada asistencia alumno "
                        + "con nia = " + row.nia)
            else:
                syslog.syslog(LOG_INFO, "asistencia sin validar mac = " + mac)

                cursor.execute(insertSTMT, mac)

        cursor.commit()

        syslog.syslog(LOG_INFO, "macs insertadas")

        subprocess.call(['rm', '-rf', filename])

        time.sleep(capture_interval)

except KeyboardInterrupt:
    pass

cursor.close()
cnxn.close()
syslog.closelog()
```

**Figura C-2: Inserción y borrado de asistencias mediante el script Python**

```
sudo airodump-ng -w $1 --write-interval $(($2)) --output-format csv mon0 > /dev/null 2>&1 &
sleep $(($3))
sudo killall airodump-ng
```

**Figura C-3: Script bash para la ejecución de *airodump-ng***

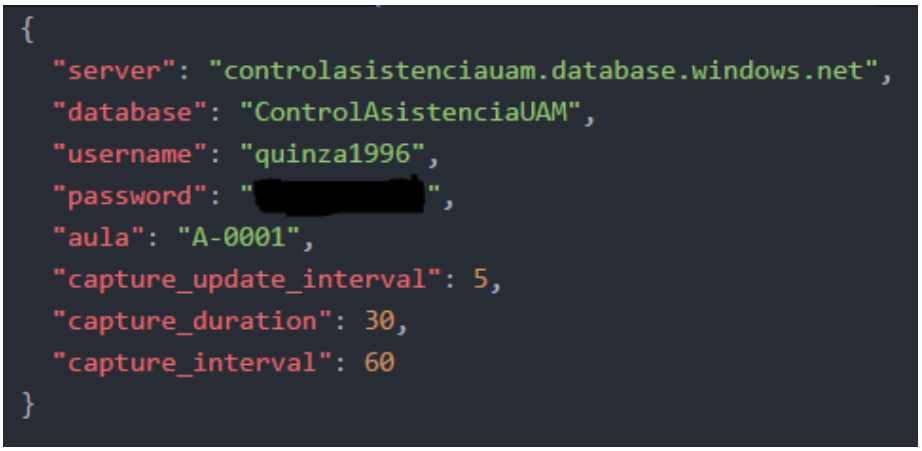


**Figura C-4: Archivo de configuración del script de la Raspberry Pi**

| | | | |
|---|---|---|---|
| Assets | 24/01/2018 13:08 | Carpeta de archivos | |
| bin | 24/01/2018 13:08 | Carpeta de archivos | |
| helpers | 04/03/2018 19:03 | Carpeta de archivos | |
| obj | 24/01/2018 16:23 | Carpeta de archivos | |
| Properties | 24/01/2018 13:08 | Carpeta de archivos | |
| Resources | 24/01/2018 13:08 | Carpeta de archivos | |
| ControlAsistencia.Android.csproj | 31/05/2018 23:36 | Visual C# Project f... | 5 KB |
| ControlAsistencia.Android.csproj.bak | 07/03/2018 21:22 | Archivo BAK | 5 KB |
| ControlAsistencia.Android.csproj.user | 13/06/2018 13:39 | Archivo de opcion... | 1 KB |
| MainActivity.cs | 24/01/2018 13:08 | Archivo de código... | 1 KB |

**Figura C-5: Estructura de carpetas dentro del proyecto de cada plataforma**

| | | | |
|---|---|---|---|
| clases | 10/04/2018 21:34 | Carpeta de archivos | |
| interfaces | 04/03/2018 19:01 | Carpeta de archivos | |
| Recursos | 03/04/2018 18:07 | Carpeta de archivos | |
| VISTAS | 04/03/2018 15:36 | Carpeta de archivos | |
| App.xaml | 24/01/2018 13:08 | Archivo de marca... | 1 KB |
| App.xaml.cs | 15/04/2018 17:02 | Archivo de código... | 1 KB |
| ControlAsistencia.projitems | 27/05/2018 13:04 | Archivo PROJITEMS | 5 KB |
| ControlAsistencia.shproj | 24/01/2018 13:08 | Code Sharing App... | 1 KB |

**Figura C-6: Estructura de carpetas en el proyecto compartido**

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ControlAsistencia.clases
{
    58 referencias | Javier Quinzaños, Hace 15 días | 1 autor, 1 cambio
    public static class Constants
    {
        /** Para los parametros de conexión a base de datos */
        public const String DB_USERNAME = "quinza1996";
        public const String DB_PASSWORD = "[redacted]";
        public const String DB_NAME = "ControlAsistenciaUAM";
        public const String DB_SERVER = "controlasistenciauam.database.windows.net";

        /** Para indicar los minutos que pueden transcurrir desde el inicio de la clase para aceptar asistencias */
        public const byte MINUTES_ASSIST = 15;

        /** Titulos de cada página */
        public const String REGISTER_TITLE = "Registro";
        public const String CONFIRM_ATT_TITLE = "Confirmar asistencia";
        public const String GET_ATT_TITLE = "Abrir asistencias";
        public const String SETTINGS_TITLE = "Ajustes";
        public const String CHECK_ATT_TITLE = "Comprobar asistencias";

        /** Para saber la terminación del correo electronico de cada usuario */
        public const String TEACHER_MAIL_END = "uam.es";
        public const String STUDENT_MAIL_END = "estudiante." + TEACHER_MAIL_END;

        /** Indica si es profesor o alumno */
        public const String USER_TYPE_KEY = "esProfesor";

        /** Para datos del alumno */
        public const String NIA_KEY = "nia";
        public const String MAC_KEY = "mac";

        /** Para datos del profesor */
        public const String EMAIL_KEY = "email";

        /** Nombre de los archivos JSON de la aplicacion */
        public const String USERS_FILENAME = "users";
        public const String CLASSROOMS_FILENAME = "aulas_por_dias"; // aulas
        public const String MATRICULAS_FILENAME = "matriculas";
        public const String ASIGNATURAS_FILENAME = "asignaturas";
    }
}
```

**Figura C-7: Constantes de la aplicación móvil**

```
using ControlAsistencia.vistas;
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Reflection;
using Xamarin.Forms;

namespace ControlAsistencia.clases
{
    20 referencias | Javier Quinzaños, Hace 15 días | 1 autor, 2 cambios
    public static class JSONLoader
    {
        8 referencias | Javier Quinzaños, Hace 58 días | 1 autor, 1 cambio
        public static JObject Load (String filename)
        {
            JObject json;

            var assembly = IntrospectionExtensions.GetTypeInfo(typeof(RegisterPage)).Assembly;

            String file = "Recursos." + filename + ".json";
            String assemblyName = "ControlAsistencia.";
            String path = "";

            if (Device.RuntimePlatform == Device.iOS)
                path = assemblyName + "iOS.";
            else if (Device.RuntimePlatform == Device.Android)
                path = assemblyName + "Droid.";
            else if (Device.RuntimePlatform == Device.UWP)
                path = assemblyName + "UWP.";

            path += file;

            Stream stream = assembly.GetManifestResourceStream(path);

            if (stream == null)
                return null;

            using (var reader = new StreamReader(stream))
            {
                String text = reader.ReadToEnd();
                json = JObject.Parse(text);
            }

            return json;
        }
```

**Figura C-8: Lectura de un archivo JSON en la aplicación**

```
1 referencia | Javier Quinzaños, Hace 15 días | 1 autor, 1 cambio
public static IEnumerable<UInt32> GetAsignaturasQueEmpiezanAhora (JObject json, DateTime ahora)
{
    int day = ((int) DateTime.Now.DayOfWeek + 6) % 7;

    if (day > 5)
        return Enumerable.Empty<UInt32>();

    return from aula in json["aulas"]
           from clase in aula["clases"][day]
           let auxTime = new DateTime(ahora.Year, ahora.Month, ahora.Day, ahora.Hour, ahora.Minute, ahora.Second)
           let diff = auxTime.Subtract(DateTime.Parse((String) clase["inicio"]))
           where diff.Hours == 0 && //diff.Minutes <= Constants.MINUTES_ASSIST &&
                 ahora <= DateTime.Parse((String) clase["fin"])
           select (UInt32) clase["asignatura"];
}
```

**Figura C-9: Ejemplo de un método para obtener información de un fichero JSON**

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Linq;
using System.Data.SqlClient;

namespace ControlAsistencia.clases
{
    23 referencias | Javier Quinzaños, Hace 14 días | 1 autor, 4 cambios
    public static class SQLExtensions
    {
        public static readonly string SQL_INSERT_USUARIO =
            "INSERT INTO [user](nia, mac) VALUES (@nia, @mac)";

        public static readonly string SQL_INSERT_ASISTENCIA =
            "INSERT INTO asistencias(alumno, aula, fecha) VALUES (@nia, @aula, GETDATE())";

        public static readonly string SQL_COUNT_ASISTENCIAS =
            "SELECT COUNT(*) FROM temp_macs "
             + "WHERE mac = @mac AND CONVERT(date, fecha) = CONVERT(date, GETDATE()) "
             + "AND aula = @aula";
```

**Figura C-10: Algunas sentencias SQL usadas por la aplicación**

```
10 referencias | 0 cambios | 0 autores, 0 cambios
public static SqlConnection NewInstance
    (String user = Constants.DB_USERNAME, String password = Constants.DB_PASSWORD,
     String dbName = Constants.DB_NAME, String server = Constants.DB_SERVER)
{
    SqlConnection conn = new SqlConnection();
    SqlConnectionStringBuilder connBuilder = new SqlConnectionStringBuilder
    {
        UserID = user,
        Password = password,
        InitialCatalog = dbName,
        DataSource = server,
        ConnectTimeout = 30
    };

    conn.ConnectionString = connBuilder.ConnectionString;

    return conn;
}
```

**Figura C-11: Método para conectar con la base de datos**

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ControlAsistencia.interfaces
{
    /// <summary>
    /// Interface to implement a function to open wifi settings on each platform
    /// </summary>
    6 referencias | Javier Quinzaños, Hace 101 días | 1 autor, 1 cambio
    public interface IOpenWifiSettings
    {
        /// <summary>
        /// Opens wifi settings on the device
        /// </summary>
        4 referencias | Javier Quinzaños, Hace 101 días | 1 autor, 1 cambio
        void OpenWifiSettings();
    }
}
```

**Figura C-12: Interfaz para abrir los ajustes Wifi**

```
[assembly: Dependency(typeof(OpenWifiSettingsHelper))]
namespace ControlAsistencia.Droid.helpers
{
    using System;
    using Android.Content;
    using ControlAsistencia.interfaces;

    /// <summary>
    /// Implements <see cref="IOpenWifiSettings"/> to get into Wifi settings quickly on Android
    /// </summary>
    1 referencia | Javier Quinzaños, Hace 63 días | 2 autores, 2 cambios
    public class OpenWifiSettingsHelper : IOpenWifiSettings
    {
        /// <summary>
        /// Opens wifi settings on this Android device
        /// </summary>
        4 referencias | Javier Quinzaños, Hace 63 días | 2 autores, 2 cambios
        public void OpenWifiSettings()
        {
            try
            {
                using (var intent = new Intent(Android.Provider.Settings.ActionWifiSettings))
                {
                    intent.SetFlags(ActivityFlags.NewTask);
                    Android.App.Application.Context.StartActivity(intent);
                }
            }
            catch (Exception ex)
            {
                throw new InvalidOperationException("Opening Wifi settings was not possible. " + ex.Message);
            }
        }
    }
}
```

**Figura C-13: Funciones necesarias para abrir los ajustes Wifi en Android**

```
[assembly: Dependency(typeof(OpenWifiSettingsHelper))]
namespace ControlAsistencia.iOS.helpers
{
    using System;
    using System.Diagnostics;
    using Foundation;
    using UIKit;
    using ControlAsistencia.interfaces;

    /// <summary>
    /// Implementation of <see cref="IOpenWifiSettings"/> on iOS
    /// </summary>
    1 referencia | Javier Quinzaños, Hace 63 días | 2 autores, 2 cambios
    public class OpenWifiSettingsHelper : IOpenWifiSettings
    {
        /// <summary>
        /// Opens the wifi settings on the device
        /// </summary>
        4 referencias | Javier Quinzaños, Hace 63 días | 2 autores, 2 cambios
        public void OpenWifiSettings()
        {
            try
            {
                var wifiUrl = new NSUrl("prefs:root=WIFI");
                if (UIApplication.SharedApplication.CanOpenUrl(wifiUrl))
                {
                    // Pre iOS 10
                    UIApplication.SharedApplication.OpenUrl(wifiUrl);
                }
                else
                {
                    // iOS 10
                    using (var nSUrl = new NSUrl("App-Prefs:root=WIFI"))
                    {
                        UIApplication.SharedApplication.OpenUrl(nSUrl);
                    }
                }
            }
            catch (Exception ex)
            {
                throw new InvalidOperationException("Could not open Wifi Settings. " + ex.Message);
            }
        }
    }
}
```

**Figura C-14: Funciones necesarias para abrir los ajustes Wifi en iOS**

```
[assembly: Dependency(typeof(OpenWifiSettingsHelper))]
namespace ControlAsistencia.UWP.helpers
{
    using System;
    using ControlAsistencia.interfaces;
    using Windows.System.Profile;

    /// <summary>
    /// Implements <see cref="OpenWifiSettingsHelper"/> to open wifi settings on UWP
    /// </summary>
    2 referencias | Javier Quinzaños, Hace 101 días | 1 autor, 1 cambio
    public class OpenWifiSettingsHelper : IOpenWifiSettings
    {
        /// <summary>
        /// Opens wifi settings on this Windows device
        /// </summary>
        4 referencias | Javier Quinzaños, Hace 101 días | 1 autor, 1 cambio
        public void OpenWifiSettings()
        {
            // Open the about page when on Windows 10 Mobile, because you find the MAC address there.
            var success = AnalyticsInfo.VersionInfo.DeviceFamily == "Windows.Mobile" ?
                Windows.System.Launcher.LaunchUriAsync(new Uri("ms-settings:about")).GetAwaiter().GetResult() :
                Windows.System.Launcher.LaunchUriAsync(new Uri("ms-settings:network-wifi")).GetAwaiter().GetResult();

            if (!success)
            {
                throw new InvalidOperationException("Could not open Wifi settings");
            }
        }
    }
}
```

**Figura C-15: Funciones necesarias para abrir los ajustes Wifi en Windows**

```
{
  "profesores": [
    {
      "email": "test@uam.es",
      "asignaturas": [12345]
    },
    {
      "email": "alvaro.ortigosa@uam.es",
      "asignaturas": []
    },
    {
      "email": "fernando.maestre@uam.es",
      "asignaturas": []
    },
    {
      "email": "pablo.castells@uam.es",
      "asignaturas": [18774]
    }
  ],
  "alumnos": [
    {
      "email": "test@estudiante.uam.es",
      "nia": 123456,
      "asignaturas": [12345, 18774]
    },
    {
      "email": "javier.quinzannos@estudiante.uam.es",
      "nia": 320098,
      "asignaturas": [18774]
    }
  ]
}
```


**Figura C-16: Fichero JSON con los posibles usuarios de la aplicación**


```
{
  "matriculas": [
    {
      "asignatura": 18774,
      "alumnos": [320098, 123456]
    },
    {
      "asignatura": 12345,
      "alumnos": [123456, 654321]
    }
  ]
}
```


**Figura C-17: Fichero JSON con los alumnos matriculados en cada asignatura**


```
{
  "asignaturas": [
    {
      "id": 12345,
      "abrev": "ATEST",
      "nombre": "Asignatura de prueba",
      "profesor": "test@uam.es"
    },
    {
      "id": 18774,
      "abrev": "BMINF",
      "nombre": "Busqueda y minería de la información",
      "profesor": "pablo.castells@uam.es"
    }
  ]
}
```


**Figura C-18: Fichero JSON con la información de las asignaturas**


**Figura C-19: Fichero JSON con los horarios y aulas de cada asignatura**

```
namespace ControlAsistencia
{
    9 referencias | 0 cambios | 0 autores, 0 cambios
    public class Principal : MasterDetailPage
    {
        private MenuPrincipalMaster MasterPage;

        private static Principal instance;

        1 referencia | 0 cambios | 0 autores, 0 cambios
        private Principal ()
        {
            MasterPage = MenuPrincipalMaster.getInstance();
            MasterPage.ListView.ItemSelected += ListView_ItemSelected;

            Master = MasterPage;

            FillInitialDetailPage();
        }

        2 referencias | 0 cambios | 0 autores, 0 cambios
        public static Principal NewInstance()
        {
            if (Principal.instance == null)
                Principal.instance = new Principal();

            return Principal.instance;
        }

        2 referencias | 0 cambios | 0 autores, 0 cambios
        public void FillInitialDetailPage()
        {
            if (!Application.Current.Properties.ContainsKey(Constants.USER_TYPE_KEY))
            {
                Detail = new NavigationPage(new RegisterPage());
            }
            else
            {
                bool esProfesor = (bool) Application.Current.Properties[Constants.USER_TYPE_KEY];

                if (esProfesor)
                {
                    Detail = new NavigationPage(new CheckAttendancePage());
                }
                else
                {
                    Detail = new NavigationPage(new ConfirmAttendancePage());
                }
            }
        }

        1 referencia | 0 cambios | 0 autores, 0 cambios
        private void ListView_ItemSelected(object sender, SelectedItemChangedEventArgs e)
        {
            var item = e.SelectedItem as MenuPrincipalMenuItem;
            if (item == null)
                return;

            var page = (Page) Activator.CreateInstance(item.TargetType);
            page.Title = item.Title;

            Detail = new NavigationPage(page);
            IsPresented = false;

            MasterPage.ListView.SelectedItem = null;
        }
    }
}
```

**Figura C-20: Punto de entrada de la aplicación**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="ControlAsistencia.vistas.ConfirmAttendancePage">

    <StackLayout>

        <Grid x:Name="confirmLayout"
              Padding="60, 200, 60, 20"
              IsVisible="True">

            <Grid.RowDefinitions>
                <RowDefinition Height="6*"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>

            <StackLayout Grid.Row="0" Spacing="40">
                <Label x:Name="txtInfoAsistencia"
                       HorizontalOptions="CenterAndExpand"
                       FontSize="Medium"/>
                <Label x:Name="txtAsignatura"
                       HorizontalOptions="CenterAndExpand"
                       FontAttributes="Bold"
                       FontSize="Large"
                       TextColor="DeepSkyBlue"/>
                <Label x:Name="txtAula"
                       Text="en el aula "
                       HorizontalOptions="CenterAndExpand"
                       FontSize="Medium"/>
            </StackLayout>

            <StackLayout Grid.Row="1">
                <Button x:Name="btnConfirm"
                    HorizontalOptions="CenterAndExpand"
                    VerticalOptions="EndAndExpand"
                    FontSize="Medium"
                    BorderWidth="5"
                    BorderColor="LightGray"
                    BorderRadius="15"
                    Clicked="btnConfirm_Clicked"/>
            </StackLayout>

        </Grid>

        <StackLayout x:Name="errorLayout"
                 Padding="40, 0, 40, 0"
                 IsVisible="False"
                 VerticalOptions="CenterAndExpand"
                 HorizontalOptions="Center">

            <Label x:Name="txtError"
                TextColor="DarkRed"
                FontSize="Medium"/>

        </StackLayout>

    </StackLayout>

</ContentPage>
```

**Figura C-21: Definición de la interfaz para confirmar las asistencias**

```
using ControlAsistencia.clases;
using Newtonsoft.Json.Linq;
using System;
using System.Linq;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Threading.Tasks;

namespace ControlAsistencia.vistas
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    10 referencias | 0 cambios | 0 autores, 0 cambios
    public partial class ConfirmAttendancePage : ContentPage
    {
        2 referencias | 0 cambios | 0 autores, 0 cambios
        private DateTime Fecha { get; set; }
        5 referencias | 0 cambios | 0 autores, 0 cambios
        private UInt32 IdAsignatura { get; set; }
        6 referencias | 0 cambios | 0 autores, 0 cambios
        private string Aula { get; set; }

        private bool asistencia = false;

        1 referencia | 0 cambios | 0 autores, 0 cambios
        public ConfirmAttendancePage ()
        {
            InitializeComponent ();
        }

        2 referencias | 0 cambios | 0 autores, 0 cambios
        async protected override void OnAppearing()
        {
            base.OnAppearing();

            await UpdateUI();
        }

        1 referencia | 0 cambios | 0 autores, 0 cambios
        async Task UpdateUI () {
            bool esProfesor;

            if (!Application.Current.Properties.ContainsKey(Constants.USER_TYPE_KEY))
                esProfesor = false;
            else
                esProfesor = (bool) Application.Current.Properties[Constants.USER_TYPE_KEY];

            if (esProfesor)
                Title = Constants.GET_ATT_TITLE;
            else
                Title = Constants.CONFIRM_ATT_TITLE;

            DateTime ahora = DateTime.Now;

            this.Fecha = ahora;
```

**Figura C-22: *Code Behind* de la página *ConfirmAttendance* (parte 1/3)**

```
JObject jsonAsignaturasUsuario = JSONLoader.Load(Constants.USERS_FILENAME);

if (jsonAsignaturasUsuario == null)
{
    System.Diagnostics.Debug.WriteLine("Error: Ha ocurrido un error abriendo el recurso \""
        + Constants.USERS_FILENAME + ".json\".");
    return;
}

IEnumerable<UInt32> asignaturasUsuario;

if (esProfesor)
{
    if (asistencia)
    {
        txtInfoAsistencia.Text = "Deseas deshabilitar la ";
        btnConfirm.Text = "Deshabilitar ";
    }
    else
    {
        txtInfoAsistencia.Text = "Deseas habilitar la ";
        btnConfirm.Text = "Habilitar ";
    }


    UInt32 nia;
    if (!Application.Current.Properties.ContainsKey(Constants.NIA_KEY))
        nia = 123456;
    else
        nia = (UInt32)Application.Current.Properties[Constants.NIA_KEY];

    asignaturasUsuario = JSONLoader.GetAsignaturasAlumno(jsonAsignaturasUsuario, nia);
}
else
{
    txtInfoAsistencia.Text = "Deseas confirmar tu ";
    btnConfirm.Text = "Confirmar ";

    String email;
    if (!Application.Current.Properties.ContainsKey(Constants.EMAIL_KEY))
        email = "test@uam.es";
    else
        email = (string) Application.Current.Properties[Constants.EMAIL_KEY];

    asignaturasUsuario = JSONLoader.GetAsignaturasProfesor(jsonAsignaturasUsuario, email);
}

txtInfoAsistencia.Text += "asistencia a la asignatura:";
btnConfirm.Text += "asistencia";
```

**Figura C-23: *Code Behind* de la página *ConfirmAttendance* (parte 2/3)**

```
JObject jsonAulas = JSONLoader.Load(Constants.CLASSROOMS_FILENAME);

if (jsonAulas == null)
{
    System.Diagnostics.Debug.WriteLine("Error: Ha ocurrido un error abriendo el recurso \""
        + Constants.CLASSROOMS_FILENAME + ".json\".");
    return;
}

var asignaturasQueEmpiezan = JSONLoader.GetAsignaturasQueEmpiezanAhora(jsonAulas, ahora);

comment3

UInt32? asignatura = (from e in asignaturasQueEmpiezan
                      where asignaturasUsuario.Contains(e)
                      select e).Cast<UInt32?>().FirstOrDefault();

if (asignatura == null)
{
    System.Diagnostics.Debug.WriteLine("INFO: No se ha encontrado ninguna asignatura de ese alumno o profesor a esta hora.");

    this.confirmLayout.IsVisible = false;
    this.errorLayout.IsVisible = true;
    this.txtError.Text = "No tienes ninguna clase ahora mismo.";

    return;
}

this.IdAsignatura = asignatura.Value;

bool puedeAsistir = await IsAttendanceOpen(this.IdAsignatura, ahora);

if (puedeAsistir || esProfesor)
{
    JObject jsonAsignaturas = JSONLoader.Load(Constants.ASIGNATURAS_FILENAME);

    if (jsonAsignaturas == null)
    {
        System.Diagnostics.Debug.WriteLine("Error: Ha ocurrido un error abriendo el recurso \""
            + Constants.ASIGNATURAS_FILENAME + ".json\".");
        return;
    }

    this.txtAsignatura.Text = JSONLoader.GetNombreAsignaturaByID(jsonAsignaturas, asignatura.Value);

    this.Aula = JSONLoader.GetAulaAsignaturaPorIDyFecha(jsonAulas, asignatura.Value, ahora);
    this.txtAula.Text += this.Aula;

    this.errorLayout.IsVisible = false;
    this.confirmLayout.IsVisible = true;
}
else
{
    this.confirmLayout.IsVisible = false;
    this.errorLayout.IsVisible = true;
    this.txtError.Text = "No puedes confirmar tu asistencia. El profesor no lo ha activado.";
```

**Figura C-24: *Code Behind* de la página *ConfirmAttendance* (parte 3/3)**

```
2 referencias | 0 cambios | 0 autores, 0 cambios
private Task<bool> IsAttendanceOpen(UInt32 idAsignatura, DateTime fecha)
{
    var conn = SQLExtensions.NewInstance();

    bool res = false;

    try
    {
        conn.Open();

        using (SqlCommand comando = new SqlCommand(SQLExtensions.SQL_SELECT_HABIL_ASIST, conn))
        {
            comando.Parameters.Add("@asignatura", SqlDbType.Int, 5).Value = idAsignatura;
            comando.Parameters.Add("@fecha", SqlDbType.DateTime).Value = fecha;
            comando.CommandType = CommandType.Text;
            var reader = comando.ExecuteReader();

            if (reader.Read())
            {
                res = reader.GetBoolean(0);
            }
        }

        conn.Close();
    }
    catch (Exception excep)
    {
        System.Diagnostics.Debug.WriteLine("Error con la base de datos. " + excep.Message);
    }

    return Task.FromResult(res);
}
```

**Figura C-25: Método para comprobar si está abierta la asistencia**

```
0 referencias | 0 cambios | 0 autores, 0 cambios
private async void btnConfirm_Clicked(object sender, EventArgs e)
{
    bool esProfesor;

    if (!Application.Current.Properties.ContainsKey(Constants.USER_TYPE_KEY))
        esProfesor = false;
    else
        esProfesor = (bool) Application.Current.Properties[Constants.USER_TYPE_KEY];

    if (!esProfesor) // Es alumno
    {
        comment4

        /* Comprobar si esta abierta la asistencia */

        DateTime ahora = DateTime.Now;

        bool puedeAsistir = await IsAttendanceOpen(this.IdAsignatura, ahora);

        if (!puedeAsistir)
        {
            await DisplayAlert("Información!", "No está habilitada la asistencia para esta asignatura.", "OK");
            return;
        }

        if (this.asistencia)
        {
            await DisplayAlert("Informacion!", "Ya habías confirmado tu asistencia. Vuelve más tarde", "OK");
            return;
        }

        await InsertAttendanceAsync();

        this.asistencia = true;

    //////////////////////
    // Es profesor
    //////////////////////
    } else {

        if (this.asistencia)
        {
            await RejectAttendanceAsync();
        }
        else
        {
            await AcceptAttendanceAsync();
        }

        this.asistencia = !this.asistencia;
    }
}
```

**Figura C-26: *Listener* del botón que confirma (o habilita/deshabilita) la asistencia**

## *D Capturas de la aplicación*

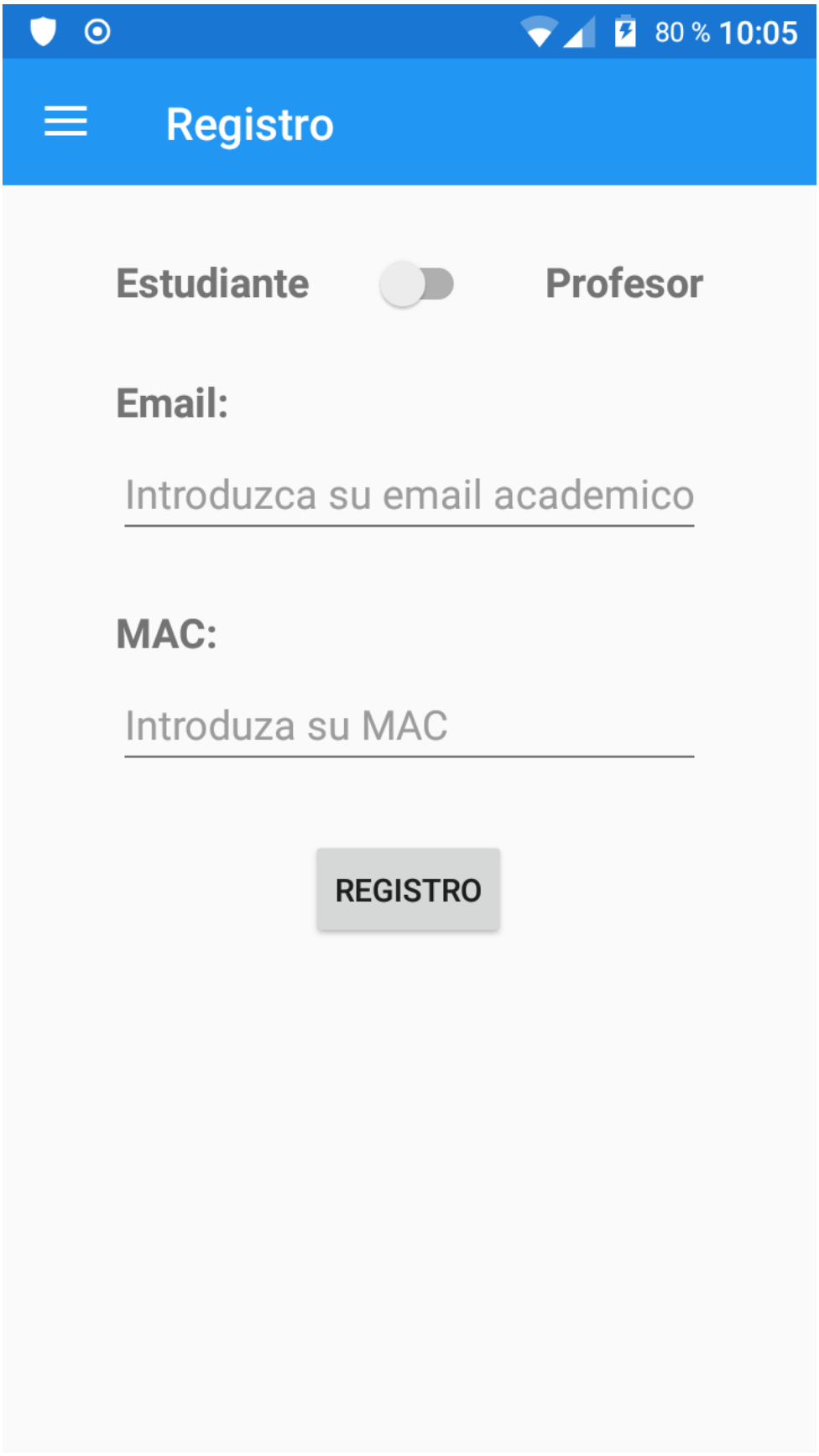


**Figura D-1: Página de registro (para un alumno)**

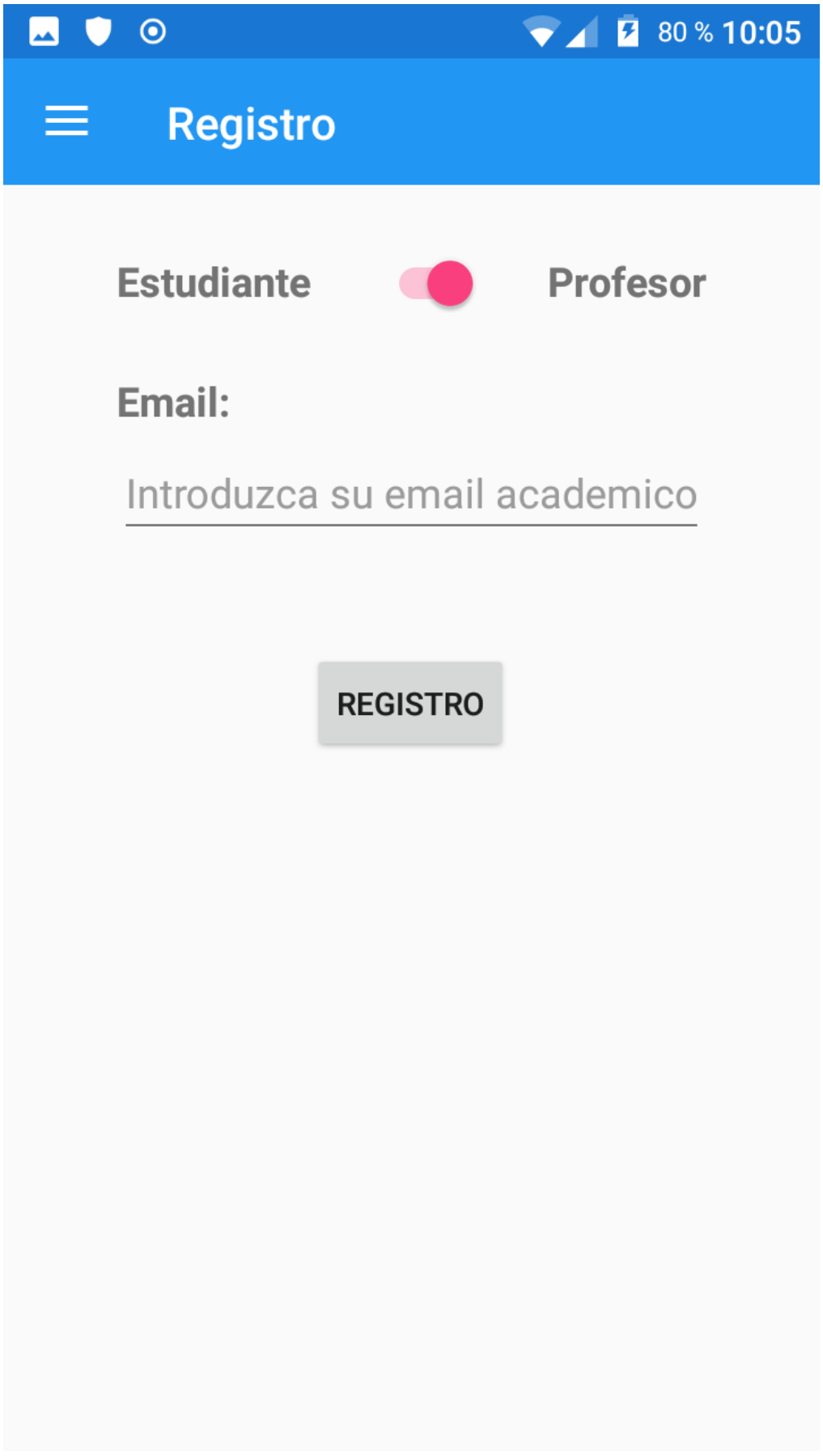


**Figura D-2: Página de registro (para un profesor)**


**Figura D-3: Menú desplegable visible por el alumno tras el registro**

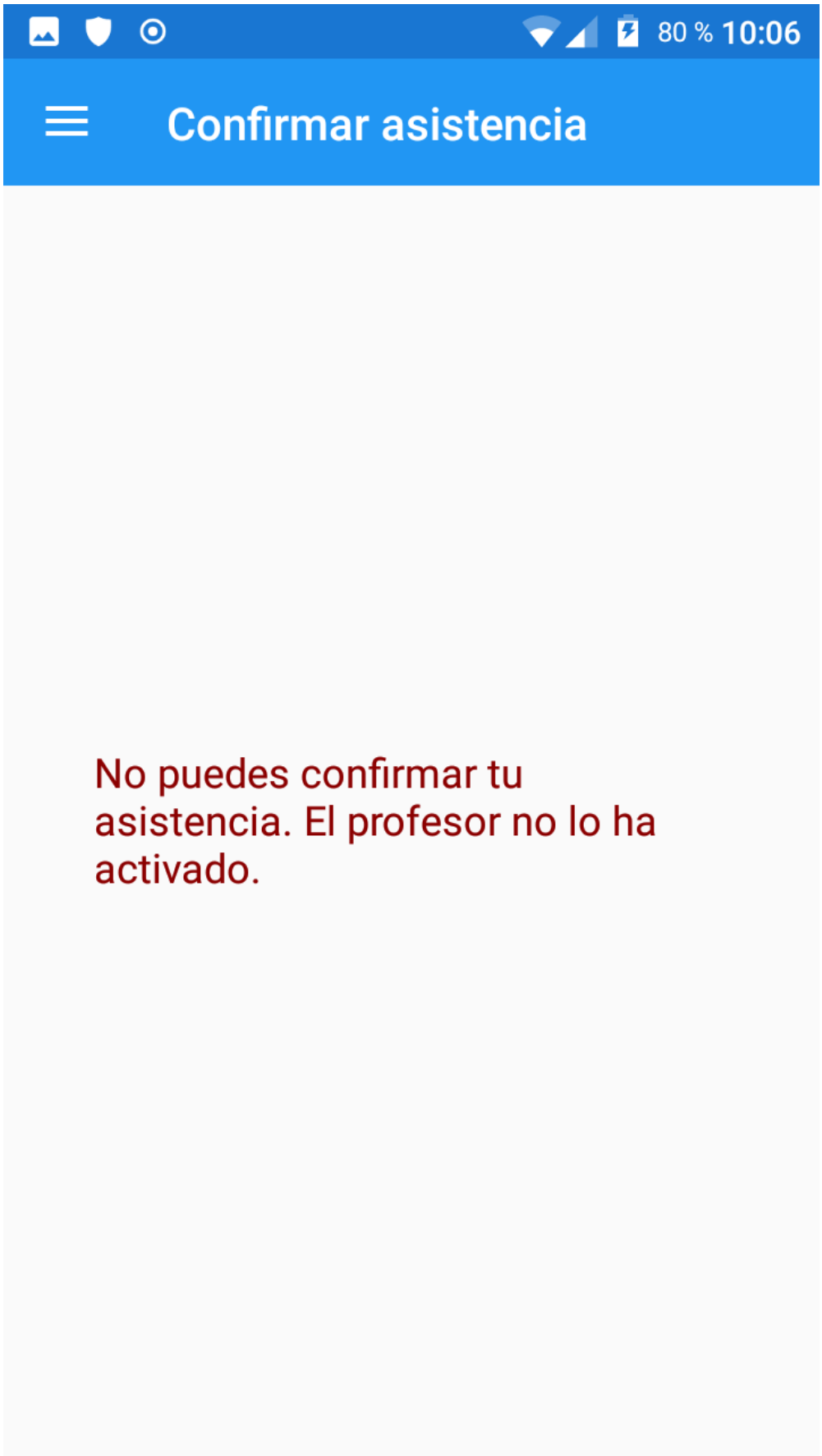


**Figura D-4: Página donde el alumno verifica su asistencia (cuando el profesor no ha habilitado el registro)**

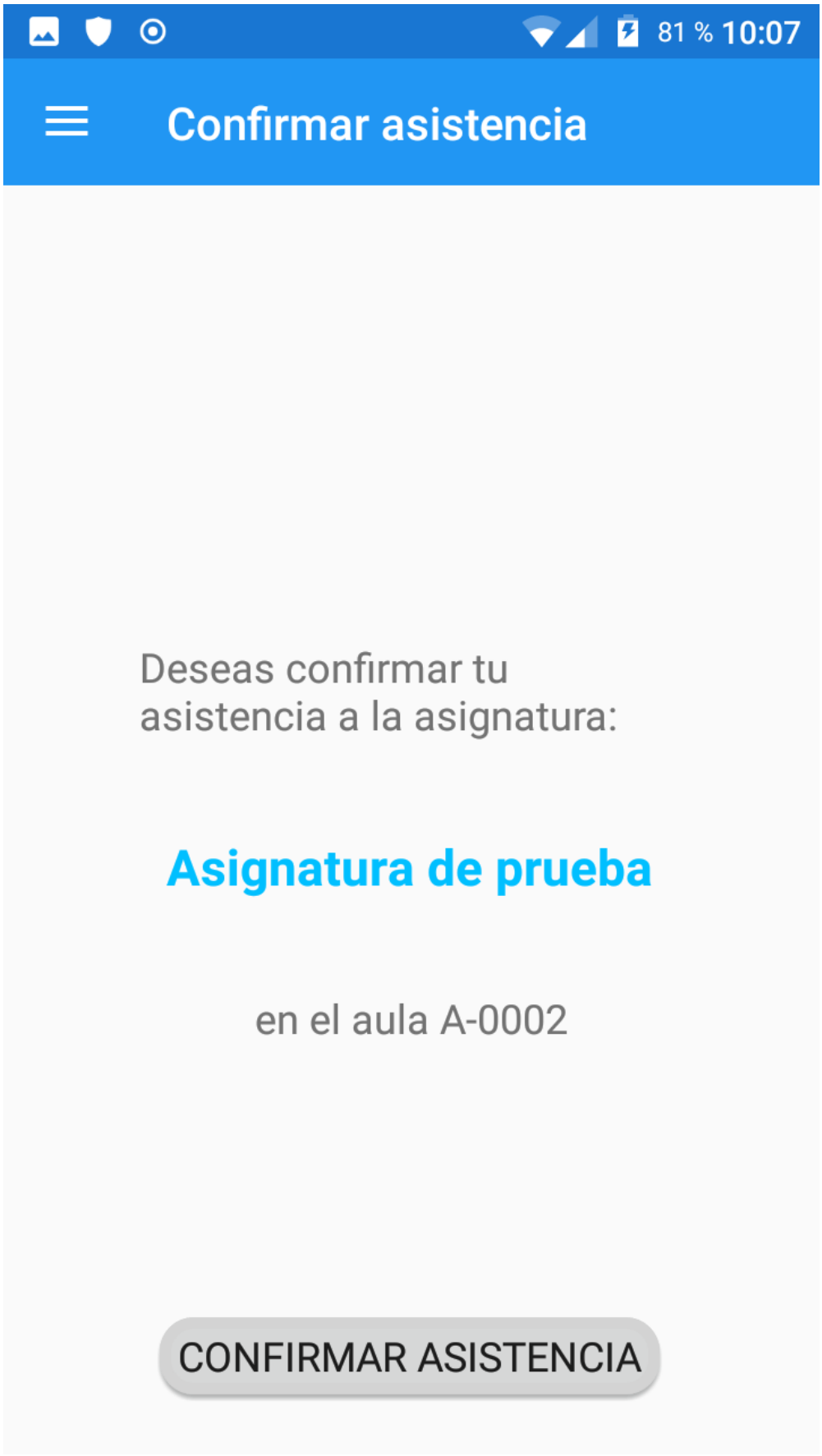


**Figura D-5: Página donde el alumno verifica su asistencia (cuando se le permite)**

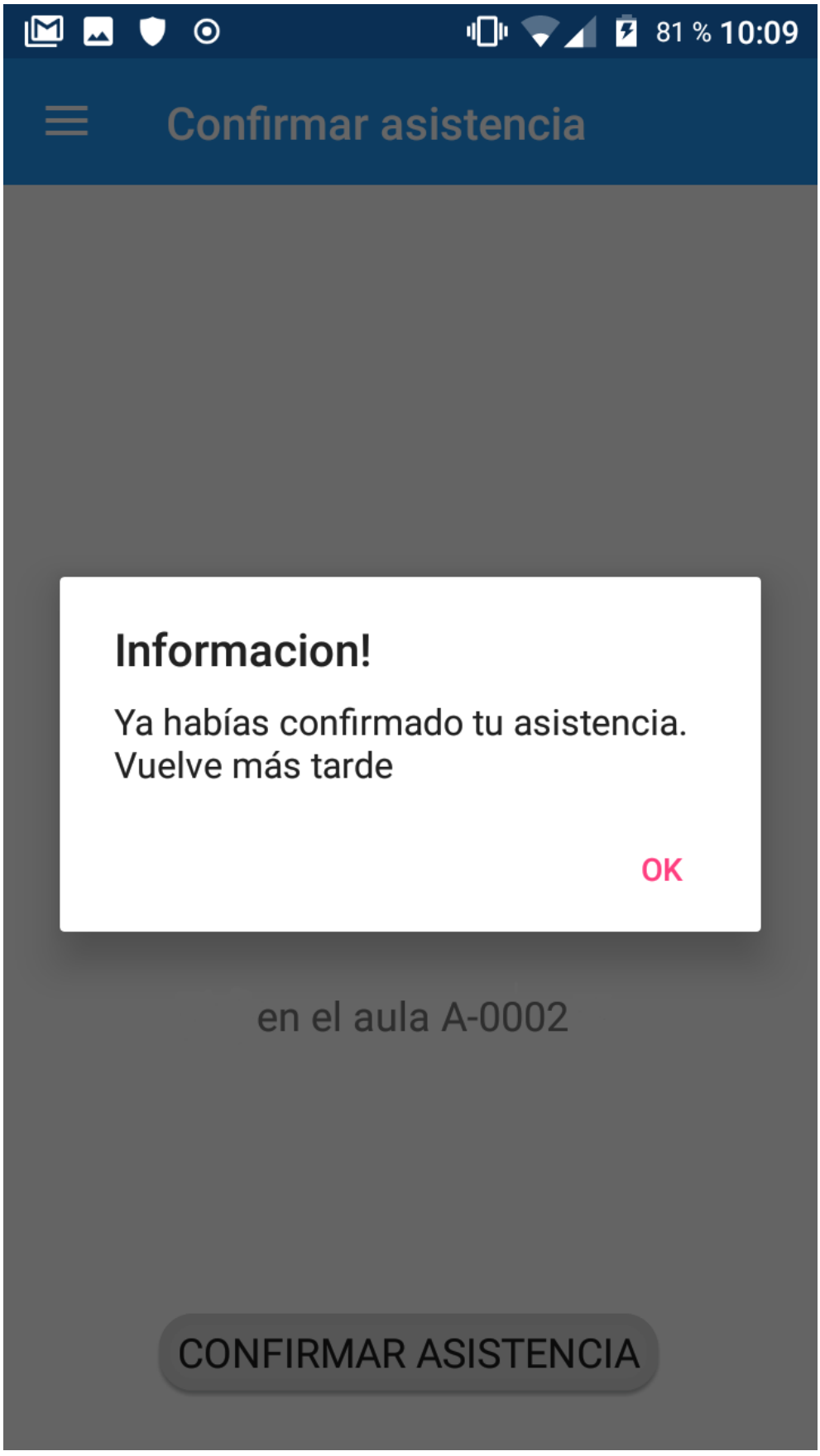


**Figura D-6: Página donde el alumno verifica su asistencia (cuando pulsa el botón de nuevo tras haber confirmado su asistencia)**

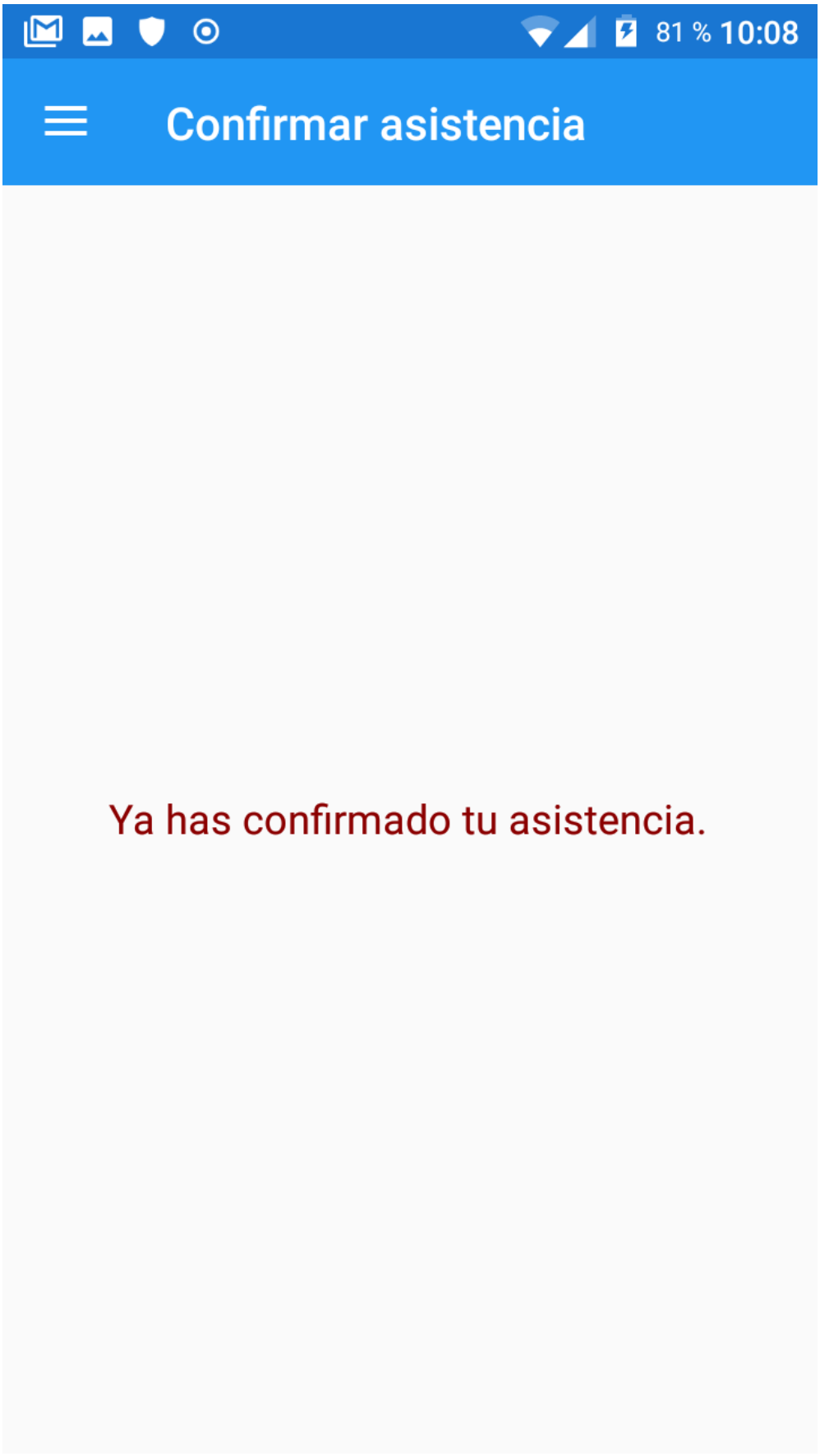


**Figura D-7: Página donde el alumno verifica su asistencia (cuando navega a otra página y vuelve, habiendo marcado ya la asistencia en una clase)**

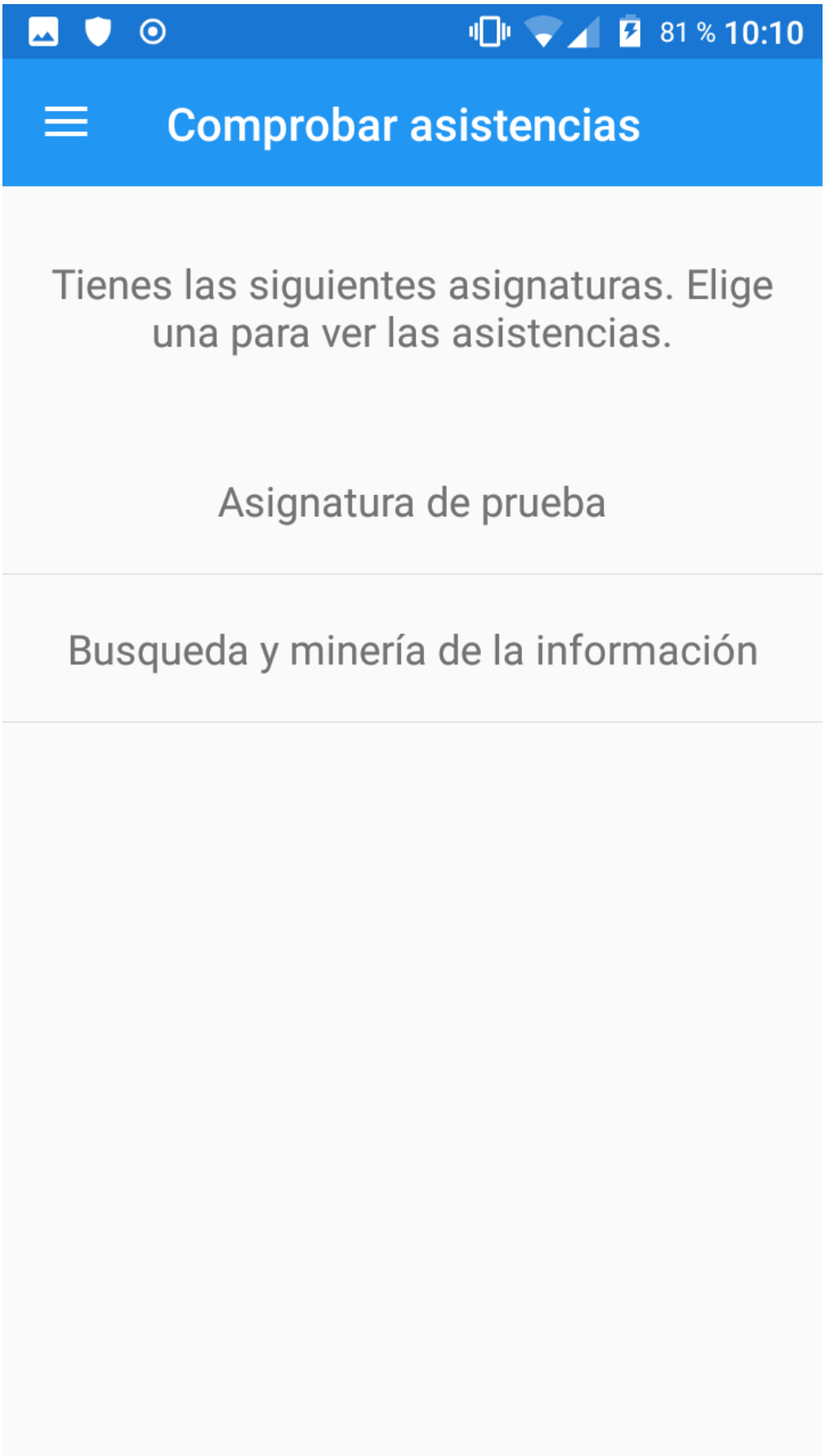


**Figura D-8: Página donde se verifican las asistencias (paso de elegir asignatura)**

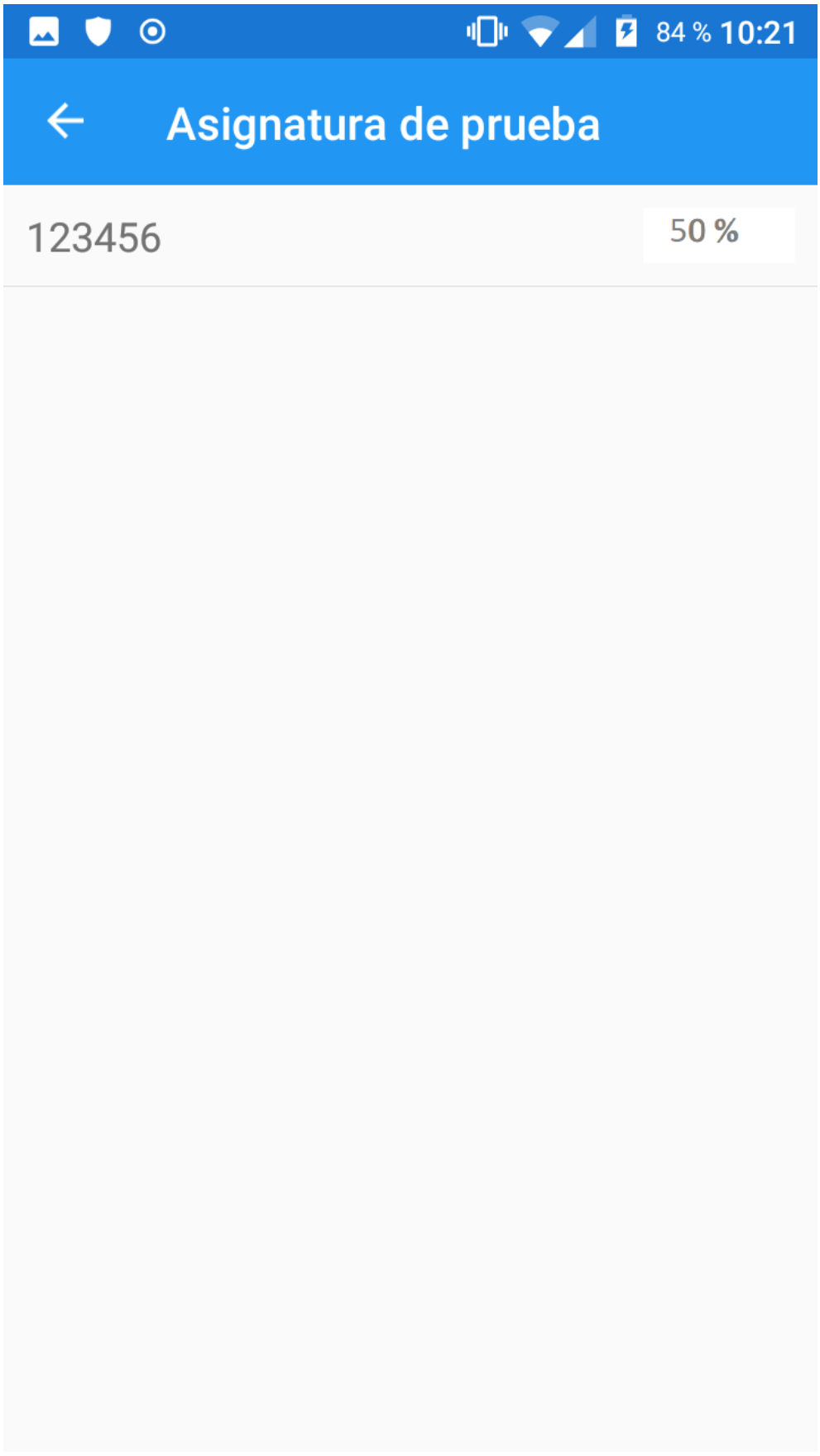


**Figura D-9: Página donde se verifican las asistencias (se muestra una lista de alumnos con su porcentaje de faltas a esa asignatura)**